Lecture Notes in Social Networks

Tansel Özyer
Zeki Erdem
Jon Rokne
Suheil Khoury *Editors*

Mining Social Networks and Security Informatics

Springer

Mining Social Networks and Security Informatics

Lecture Notes in Social Networks (LNSN)

For further volumes:
www.springer.com/series/8768

Tansel Özyer • Zeki Erdem • Jon Rokne •
Suheil Khoury
Editors

Mining Social Networks and Security Informatics

Springer

Editors
Tansel Özyer
Department of Computer Engineering
TOBB University
Sogutozu Ankara, Turkey

Zeki Erdem
Information Technologies Institute
TUBITAK BILGEM
Kocaeli, Turkey

Jon Rokne
Computer Science
University of Calgary
Calgary, Canada

Suheil Khoury
Department of Mathematics and Statistics
American University of Sharjah
Sharjah, Saudi Arabia

ISSN 2190-5428
ISSN 2190-5436 (electronic)
Lecture Notes in Social Networks
ISBN 978-94-007-6358-6
ISBN 978-94-007-6359-3 (eBook)
DOI 10.1007/978-94-007-6359-3
Springer Dordrecht Heidelberg New York London

Library of Congress Control Number: 2013939726

Cover design: eStudio Calamar, Berlin/Figueres

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Contents

A Model for Dynamic Integration of Data Sources

Murat Obali and Bunyamin Dursun

Abstract Online and offline data is the key to Intelligence Agents, but these data cannot be fully analyzed due to the wealth and complexity and non-integrated nature of the information available. In the field of security and intelligence, there is a huge number of data coming from heterogonous data sources in different formats. The integration and the management of these data are very costly and time consuming. The result is a great need for dynamic integration of these intelligent data. In this paper, we propose a complete model that integrates different online and offline data sources. This model takes part between the data sources and our applications.

Keywords Online data · Offline data · Data source · Infotype · Information · Fusion · Dynamic data integration · Schema matching · Fuzzy match

1 Introduction

Heterogonous databases are growing exponentially as in Moore's law. Data integration importance is increasing as the volume of data and the need to share this data increase.

As the years went by, most enterprise data fragmented in different data sources. So, they have to combine these data and to view in a unified form.

Online and offline data is the key to Intelligence Agents, but we cannot fully analyze this data due to the wealth and complexity and non-integrated nature of the information available [2].

In the field of security and intelligence, there is a huge number of data coming from heterogonous data sources in different formats. How to integrate and manage, and finding relations between these data are crucial points for analysis. When a new data source is added or an old data source is changed by means of data structure,

M. Obali (✉) · B. Dursun
Tubitak Bilgem Bte, Ankara, Turkey
e-mail: murat.obali@tubitak.gov.tr

B. Dursun
e-mail: bunyamin.dursun@tubitak.gov.tr

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_1,

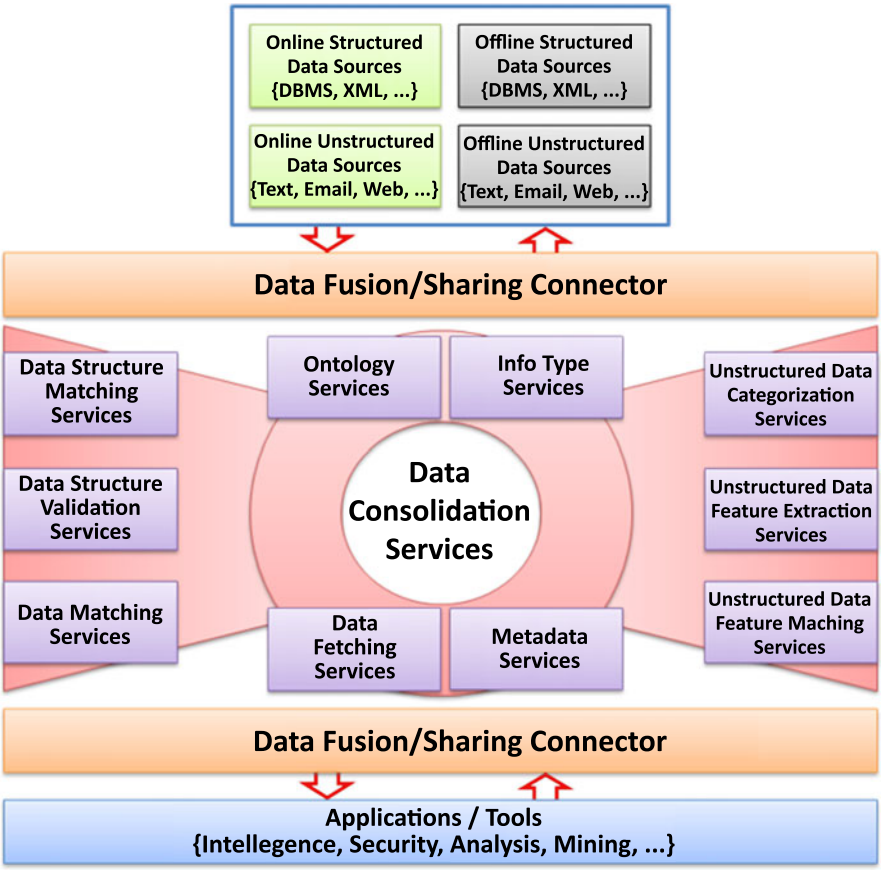


Fig. 1 General model of the system

intelligence systems which use these data sources have to change; and sometimes these changes must be made in source codes of the systems that mainly require analyzing, designing, coding, testing and deploying phases. That is loss of time and money. The result is a great need for dynamic integration of these intelligent data.

However, in many traditional approaches such as federated database systems and data warehouses; there is a lack of integration because of changing nature of the data sources [11]. In addition, continuing change and growth of data sources results in expensive and hard successive software maintenance operations [7, 9].

We propose a new conceptual model for the integration of different online and offline data sources. This model is shown in Fig. 1. Our model requires minimal changes for adapting new data sources. Any data sources and data processing systems can be attached to our model and the model provides the communication between both systems. Our model proposes a new approach called “Info Type” for matching and fetching needs.

1.1 What Is Data Integration?

Data integration is basically combining data residing at different data sources, and providing a unified view of these data [13]. This process is significant in a variety of situations and sometimes is of primary importance.

Today, data integration is becoming important in many commercial/in-house applications and scientific research.

1.2 Is Data Integration a Hard Problem?

Yes, Data Integration is a hard problem and it's not only IT people problem but also IT users' problem. First, the data in the world sometimes too complex and applications was not designed in a data integration friendly fashion. Also, application fragmentation brings about data fragmentation. We use different database systems and thus use different interfaces, different architectural designs and different file formats etc. Furthermore, the data is dirty, not in a standard format. Same words may not be same meaning and you cannot easily integrate them.

2 Data Sources

2.1 What Is Data Source?

Data Source, as the name implies provides data. Some known examples are a database, a computer file and a data stream.

2.2 Data Source Types

In this study, we categorize data into online, offline, structured and unstructured by means of their properties.

In general, "online" indicates a state of connectivity, while "offline" indicates a disconnected state. Here, we mean that online is connected to a system, in operation, functional and ready for service. In contrast, an offline data means no connection, in a media such as CD, Hard Disk or sometimes on a paper. It's important for security and intelligence to integrate offline data to improve online relevancy [4].

As the name implies, structured means well-defined formatted data such as database tables and excel spread sheets. In contrast, unstructured is not in well-defined format, free text data such as web pages and text documents.

2.3 Data Quality and Completeness

It is essential that a data source meets the data requirements of users for information. Data completeness is an indication of whether or not all the data necessary are available in the data resource.

Data quality refers to that correctness, completeness, accuracy, relevance and validity of data that is required.

Acceptable data quality and data completeness is crucial for Intelligence Agents. This is also important for the reliability of analysis and information.

3 Dynamic Integration of Data Sources

Intelligence and Warning which is identified in [5] is a mission-critical area which reports that IT researchers can help build new information and intelligence gathering and analysis capabilities to detect future illegal activities.

To consolidate data coming from different sources, data structures must match corresponding data structure. There are many algorithms to solve it [6]. In many cases, data structures must match acceptable structural items in reference tables. For example, citizenship id, tax office, tax number fields in a sales table and in a user's table must match the pre-recorded names. So, most of the techniques found in specific schema matching algorithms will be used in the system: name similarity, thesauri, common schema structure, overlapping instances, common value distribution, re-use of past mappings, constraints, similarity to standard schemas, and common-sense reasoning [3].

A significant challenge in such a scenario is to implement an efficient and accurate fuzzy match operation that can effectively clean an incoming structure item if it fails to match exactly with any structure item in the reference relation [10] shown in Fig. 2.

3.1 Data Structure Matching

Data Structure Matching Services will work on columns/attributes of structured data by using fuzzy match operation as explained in Fig. 2. In order to use the related data in the different data sources by integrating with the aim of analyzing, it is firstly necessary to found logical relation between these data. For example, the columns of the tables under the different schemas of the different databases may be related to each other. It is essentially important to identify the table fields in the source databases and to detect the related fields in the intelligence analysis and the data warehouses established for reporting.

Certain data and metadata from the databases are periodically transferred to Matching DB for Data Structure Matching. The flow of data and metadata from

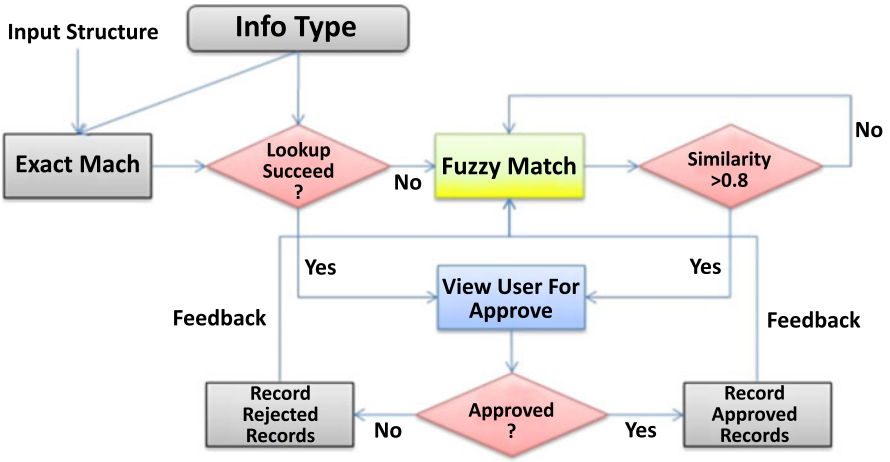
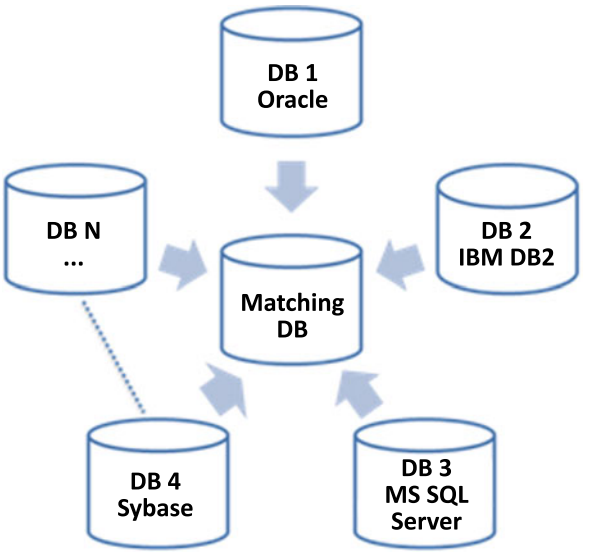


Fig. 2 Data Structure Matching

Fig. 3 Flow of the metadata from source databases



a lot of databases to Matching DB is shown in Fig. 3. The information of Database, Schema Name, Table Name and Column Name is seen in the data set transferred to Matching DB from the databases. In addition to the column information to be used for both Data Structure Matching and Data Matching, detailed information can also be provided. The additional data transferred to Matching DB from the databases are shown in Fig. 4. These additional data are discussed below:

- *Data Type:* The type of the data; numeric, character, date etc.
- *Data Length:* Maximum length that the numeric or string data fields that can take

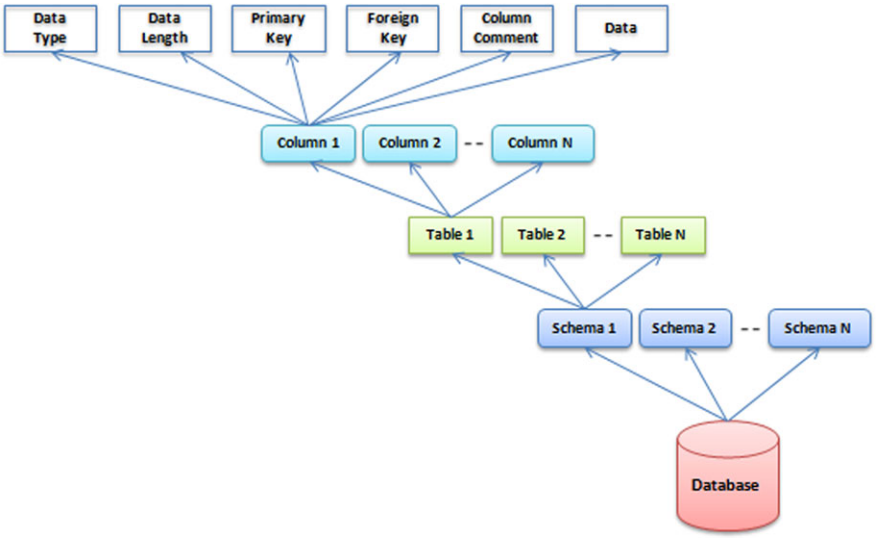


Fig. 4 The detail of the metadata coming from the source databases

- *Primary Key:* Primary keys of the tables
- *Foreign Key:* The foreign keys and reference table field information about the foreign keys
- *Column Comment:* The explanation in natural language that is inserted related to the table column by the designer of the database or developer who had created the table
- *Some Sample Data:* It is used to control the table fields matched by using different methods or to form a matching suggestion list based on the similarity of the values in the columns that couldn't be matched by using metadata.

In time, matched data sources structures may change. So we need *Data Structure Validation Services* for detecting the changes and forward them to *Data Structure Matching Services*.

Data Structure Validation Service connects to the source databases by way of related adapters in order to read the changed metadata and the sample data about the changed metadata and then writes these data to the *Matching DB* under the *Data Structure Matching Services*. The change at the source databases is monitored in here, so the new matching candidates and deletion of the old matching that became invalid is managed here.

Data Matching Services will work on data of which their structures are matched using *Data Structure Services*. In these services, 3 matching methods will be used: (1) Exact matching, (2) lookup matching and (3) functional matching.

Exact Matching means the fact that two data values are same. Because of the fact that the metadata is in uppercase in some databases such as Oracle and the metadata can be in uppercase or lowercase in some databases such as MS SQL Server, the metadata strings (for example the name of the table columns) are converted to

ASCII uppercase before the exact matching. In this way the variety caused by case sensitiveness or natural language setting removed for the advanced matching operations.

Lookup Matching means that lookup data source contains data value such as code-value pairs. Lookup Matching is used for the relations that are in the similar form of foreign keys. A table field value that is stored as a code may be related with another table field data stored not in code but in value form.

Functional Matching means comparing the data using pre-defined functions such as string similarity functions. As the different databases may be structured by different people according to different standards, and different choices for naming schema, table and column may be made, exact matching directly by metadata may lead to lose many possible matches. Therefore, even if the names of table or column are different from each other more advanced approaches for more structure matching are required. For example, matching may be made by using Edit Distance Similarity or Regular Expressions. Certain example cases for structure matching of different databases are listed below:

- *Column Name Text Similarity:* It is valid in case of the fact that there is a difference in one character of the names of two columns or the text similarity of column names is bigger than 90 %.
- *Column Name Numerator:* It means that the columns match if there are numbers as numerator at the end of the column names. For example, TELNO1, TELNO2 etc. As column names such as generic $C1, C2, \ldots, Cn$ may be used instead of the column names in certain data warehouse applications, for this kind of matching it may be added as a condition that the length of the column name is at least composed of two characters except for the numerator value at the end.
- *The matching of the column names such as X_ID and X_ NO:* ID and NO expressions at the end of column names may substitute each other while naming tables and columns. For example, a column named as OGRENCI_ID may come as OGRENCI_NO. The fact that it may be OGRENCIID and OGRENCINO without underline "_" between the words for OGRENCI_ID or OGRENCI_NO may be taken into the account in matching.
- *The matching of the column names such as X# in place of X_NO:* While naming tables and columns, # character may be used in place of NO expression at the end of the column names. For example, a column named as OGRENCI_NO may come as OGRENCI#. NO expression at the end of the column name may have been added to the previous word with or without underline.
- *The matching of the column names such as X# in place of X_ID:* While naming tables and columns, # character may be used in place of ID expression at the end of the column names. For example, a column named as OGRENCI_ID may come as OGRENCI#. ID expression at the end of the column name may have been added to the previous word with or without underline.
- *Foreign Key Relations:* As Data Structure Matching Services will be used for matching the columns in different databases, the reference columns matching according to Foreign Keys from the source databases should be included in column

Table 1 Matching dictionary table

TERM_1	LANG_1	TERM_2	LANG_ 2	TYPE
OGRENCI	TR	STUDENT	EN	Turkish-English
OGRENCI	TR	TALEBE	TR	Turkish-Turkish Synonym/Homonym
CAR	EN	VEHICLE	EN	English-English Synonym/Homonym

matching in the system used for matching. So, column pair connected to each other by foreign keys will be added to the matching as automatic query generation and auto-capture and etc. will be used in the analyses.

- *The matching suitable for "Table 1 name + Column name = Column name 2":* It means that column matching is performed in case of the fact that the text composed of the combination of table name and column name is equal to another column name. Table name and column name may have been combined directly or with an underline "_" between the column name. Supposing that there is ID column on OGRENCI table, and there is OGRENCI_ID column on OGRENCI_DERS; when the table name OGRENCI is combined with the column name ID with an underline between them, the expression OGRENCI_ID is formed. this expression is matched with the column OGRENCI_ID on the table OGRENCI_DERS. This kind of matching is usually used in case of the fact that foreign key is not performed on the database but used accordingly.
- *Dictionary Matching:* While matching the schemas the followings should be taken into the account for the words of table or column names;

  1. the choice of foreign words in naming. For example, Turkish or English word choice. For example; MUSTERI – CUSTOMER, TARIH – DATE pairs etc.
  2. using English synonym or homonym words in place of each other. For example; CAR – VEHICLE etc.

For matching by using dictionary, word pairs formed for each three cases are united on a matching dictionary table with 5 columns like in Table 1.

Matching for different languages can be carried out by this kind of table. Matching for any two languages is possible by entering the related data pairs.

- *Matching based on Table Column Comments:* System view or tables that keep the user's comment information of table columns on the databases may be used in column matching. The comments on the table columns are usually composed of a few words entered in natural language by the users and related to the meaning of the column and how it is used. According to this, the comment text the user entered is divided into its tokens, and is matched with the other table columns that have the similar names with the tokens in the text.
- *Intervention of another word between the words of the column name:* The fact that one of the pieces of the column name composed of a few pieces divided by an underline may be missing should be considered in matching. For example; OGRENCI_DERS_NOT or OGRENCI_NOT.

- *Abbreviation of the words of the column name:* The fact that one of the pieces of the column name composed of a few pieces divided by an underline may be abbreviated should be considered in matching. For example; NUFUS_KAYIT_ILCE or NUF_KAY_ILCE.
- *Combination of the words of the column name by an underline or directly:* the column names composed of a lot of pieces can be combined by an underline or directly. For example; OGRENCINOT or OGRENCI_NOT.

It is needed to run automatic and manual processes together in order to establish logical relations of data. Automatic services present the user new matching suggestions for approval. Some of these matching suggestions formed in background especially by using Functional Matching are approved or rejected by using related interfaces. While the approved matching is kept in a list as a definite relation, the ones rejected are kept in a reject list and not brought to the user again.

Some sample data with metadata are read from the source databases. This sample data is in the form of 1000 random value for each table field. For the tables that include records less than 1000, readings as much as the records on the table are made for code tables. For the pairs of table field investigated 1000 values from both of the tables are chosen. It is regarded that there are common values among these 100 values or not on both of the tables. A data similarity point depending on the numbers of common values is accounted. This data similarity point is presented to the user as additional information for approval or rejection.

Data similarity point is accounted in order for the user to ease to decide about the column pairs added to the matching candidate list by using the different matching methods above. While Accounting the similarity point, 1000 pieces of column value from the related and non empty tables are taken. This accounting is also a measurement about the fact that how many of 1000 values of one column are seen in another column. So, it is provided not to make a matching if there are outlier data even if the column names are similar. In place of sqlin below, sql with IN or EXISTS may be written. but, this is not preferred as sql will run long on big tables without index.

3.2 Unstructured Data Categorization

Unstructured data constitutes about considerable amount of the data collected or stored. Data categorization is converting the unstructured data in actionable form. That is, uncertainty to certainty, an understanding of the data on hand. This is highly necessary to manage the unstructured data [8].

Unstructured Data Categorization Services will use text mining and machine learning algorithms to categorize the unstructured data. So, most of the techniques found in specific text mining will be used in the system: text categorization, text clustering, sentiment analysis, document summarization.

3.3 Unstructured Data Feature Extraction

Transforming the unstructured data into small units (set of features) is called feature extraction. Feature extraction is an essential pre-processing step and it is often decomposed into feature construction and feature selection. To detect features are significantly important for data integration.

Unstructured Data Feature Extraction Services will work on categorized unstructured data and extract data features by using feature selection methods such as concept/entity extraction.

3.4 Unstructured Data Matching

Unstructured Data Matching Services will work on selected features of unstructured data by using fuzzy match operation as explained in Fig. 2. For fuzzy match operations, several text similarity algorithms both standard (such as Levenshtein edit distance, Jaro-Winkler similarity) and novel will be tested in order to achieve the best results.

3.5 Ontology

Ontology Services will work with ontologies recorded by user and user can search data using these ontologies. By using predefined domain ontologies such as intelligence ontologies or *foaf (friend of a friend)* format that contains human-relation information are used for detecting the annotated texts and Named Entities, and for retrieving usable data from free texts written in natural language [1, 12].

Ontologies can be used for Data Structure Matching and Data Matching. While naming the tables or table columns, preferring the synonym of the same word, using the homonym or preferring more specific or more general concepts as the column name or a piece of the column name cause not to be able to match the table columns that may be related to each other by Exact Matching or Fuzzy String Similarity methods. The quality of Data Structure Matching can be increased by using domain or global ontologies, especially by using "*is a*" and "*has a*" relations.

Ontologies can be used for matching the values in the fields that are considered to be related to each other after Data Structure Matching for *Data Matching* processes. For example, while one value in ROL field is "Manager" for one person in a human sources application, value in the related ROL column on a different database may be seen as "Director". In the cases of the fact that this kind of synonyms or hierarchic concepts can be used in place of each other, pre-defined domain ontologies should be used for Data Matching. For the unstructured data to be classified annotation can be used.

3.6 Data Matching

Info Type Services will be used for defining info types and labeling the data items. Data items coming from different data sources are mapped to these info types. Although matching of data in the fields related to each other as metadata include certain concepts and approaches mentioned in Data Structure Matching, there should be approaches special to data. String similarity, regular expressions and ontologies can be used in Data Matching.

It is possible to present an approach that can be named as Info Type. Similar data are kept in the databases of different applications. In many of the institutional applications similar fields such as "Employee Register Number", "social security number", "vehicle registration plate", "Name", "Surname", "State", "province", "occupation". While naming of these fields differ according to application and database, the data they include are similar or the same. We name this kind of common fields as Info Type. For example, "Social Security Number" may have different names in different databases such as "SSN", "SOCIAL_SECURITY_NUMBER", "SocialSecurityNumber". However, they all keep data of the same Info Type ad they all have common similarities (data type, length) and limitations.

One of the advantages of Info Type approach is the fact that identifying data fields to be integrated in different data sources, if it belongs to a certain info type, as the related info type is enough. Otherwise, it should be pointed that each pair of data field is related. Automatic transfers can be provided via the same info type in the data sources integrated. So, the relations between persons, objects and events will be automatically provided by intelligent applications in the intelligence analyses, and the analysis of huge amount of graph data will be eased.

3.7 Metadata

Metadata Services will hold all the services data such as matched structures, mapping and parameters. Identifying the data sources, periodically reading of metadata such as schema, table, table field, column, comments, foreign keys in these source databases, data that controls the operations and parameters related to monitoring and managing the structural changes in source databases are generally called as meta data in Metadata Services.

3.8 Data Fusion and Sharing

Data Fetching Services are used for fetching data by using *Metadata Services* and *Data Fusion/Sharing Connector*. For data fetching, once a query requests data, we will generate new queries (query re-writing) for each system and send it to the system, later all sub-results will be consolidated. It will be also possible to query

National Citizenship Data

TCKN	NAME	SURNAME	GENDER	PLACE_OF_BIRTH	DATE_OF_BIRTH	MARITAL_STATUS

National Tax Data

TAX_NO	TCNO	NAME	FAMILY_NAME	COMPANY	PHONE_1	PHONE_2

National Company Data

COMPANY_ID	COMPANY_NAME	FOUNDATION_DATE	ADRESS	ACTIVE	...

National Vehicle Registration Data

ID	ENGINE_NO	PLATE_NO	CHASIS_NO	NAME	SURNAME	...

Organization Membership Data

MEMBER_ID	NAME	SURNAME	COMPANY	PHONE_NUMBER	ORGANIZAION_NAME

Fig. 5 Some data sources

INFO_TYPE_ID	INFO_TYPE_NAME	MIN_LENGTH	MAX_LENGTH	VALIDATION_RULE
1	TC_KIMLIK_NO	11	11	f1(x)
2	VEHICLE_PLATE	3	10	f2(x)
3	PHONE_NUMBER	7	17	f3(X)
...	...	...	...	...

Fig. 6 Info type example

unstructured data semantically such as "*Get data if the person X is related to the murdered person Y*".

In addition to these defined services, the Dynamic Integration Model can be extended by adding other plug-in services.

All the services mentioned above will use *Data Fusion/Sharing Connector* for connecting to data sources.

4 A Sample Case

Here, we demonstrate a basic sample case. In our sample case, there are five different online structured data sources which are shown in Fig. 5, mainly related to Turkish national governmental systems.

In this model, firstly the user must define info types which relate corresponding data. A basic Info Type definition is shown in Fig. 6. Some Info Types include a val-

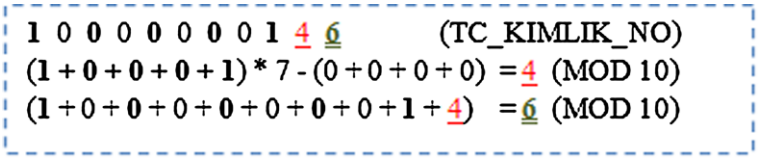


Fig. 7 TC_KIMLIK_NO validation

INFO_TYPE_ID	DATA_SOURCE_NAME	SCHEMA_NAME	DATASET_NAME	ITEM_NAME
1	National Citizenship Data	Mernis	Citizens	TCKN
1	National Tax Data	GVD	Taxes	TCNO
2	National Vehicle Registration Data	VRD	Vehicles	PLATE_NO
3	National Tax Data	GVD	Taxes	PHONE_1
3	National Tax Data	GVD	Taxes	PHONE_2
3	Organization Membership Data	OMD	Members	PHONE_NUMBER
...	...	...	...	...

Fig. 8 Info type – data item mapping

idation rule such as TC_KIMLIK_NO (Turkish Citizenship Identification Number) validation. In Turkish National Citizenship System, TC_KIMLIK_NO is validation shown in Fig. 7.

From among the data areas to be integrated, the ones that have the same info type are validated by using the same validation rules. So, both data quality is investigated for Data Integration, and more clear analyses are performed by matching using the values that are only validated before the step of Data Matching.

After *Info Types* are defined, the system connects the data sources, checks the structures and calls *Data Structure Matching Services*. *Data Structure Matching Services* maps the data items and shows user for approving. User may approve or reject the mapping, and these approve-reject records return the system as a feedback. Approved mappings are recorded to the system as shown in Fig. 8. After completing these mappings and matching, user can call *Data Fetching Services* from his/her application.

5 Conclusions and Future Work

In our sample model implementations, not only can the data be matched and get efficiently but also new data sources can be added dynamically using minimal effort. This model provides us a layer between different data sources and our applications.

So the integration of the data sources is built and managed easily. Also, proposed model is extensible and additional functions can be added.

The proposed model includes many techniques from different areas mainly machine learning, information retrieval, online-offline data sources. In future, many details will be implemented in different techniques.

References

1. Bernstein PA et al (2004) Industrial-strength schema matching. ACM SIGMOD Rec 33(4):38–43
2. Chaudhuri S et al (2003) Robust and efficient fuzzy match for online data cleaning. ACM, New York
3. Fishman A (2011) Integrating offline data to improve online relevancy. ClickZ marketing news and expert advice. [Online] Incisive interactive marketing LLC 23 03 2011. [Access Date: 17 03 2012.] http://www.clickz.com/clickz/column/2035881/integrating-offline-improve-online-relevancy
4. Heinecke J (2009) Matching natural language data with ontologies. In: Proceedings of the 4th international workshop on ontology matching, OM-2009, Chantilly, USA, October 25
5. Khan L, McLeod D (2000) Disambiguation of annotated text of audio using ontologies: ACM SIGKDD workshop on text mining, Boston
6. Lathrop RH (2001) Intelligent systems in biology: why the excitement? IEEE Intell Syst 16(6):8–13
7. Lehman MM (1996) Laws of software evolution revisited. In: Proceedings 5th European workshop on software process technology, Nancy, France
8. Lenzerini M (2002) Data integration: a theoretical perspective. In: Proceedings of the 21th ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, pp 233–246
9. Office of Homeland Security, The White House (2002) National strategy for homeland security
10. Rahm E, Bernstein PA (2001) A survey of approaches to automatic schema matching. VLDB J 10(4):334–350
11. Shariff AA, Hussain MA, Kumar S (2011) Leveraging unstructured data into intelligent information – analysis and evaluation
12. Sheth A, Larson J (1990) Federated database systems for managing distributed, heterogeneous and autonomous databases. ACM Comput Surv 22(3):183–236
13. Sood S, Vasserman L (2009) ESSE: exploring mood on the Web. In: 3rd international AAAI conference on weblogs and social media data challenge workshop

Overlapping Community Structure and Modular Overlaps in Complex Networks

Qinna Wang and Eric Fleury

Abstract In order to find overlapping community structure of complex networks, many researchers make endeavours. Here, we first discuss some existing functions proposed for measuring the quality of overlapping community structure. Second, we propose a novel algorithm called fuzzy detection for overlapping community detection. Our new method benefits from an existing partition detection technique and aims at identifying modular overlaps. A modular overlap is a group of overlapping nodes. Therefore, the overlaps shared by several communities are possibly grouped into several different modular overlaps. The results in synthetic networks and real networks demonstrate that our method can uncover and characterize meaningful overlapping nodes.

Keywords Modularity · Co-citation network · Complex networks

1 Introduction

The empirical information of networks can be used to study structural characteristics, like heavy-tailed degree distributions [1], small-world property [3] and rumour spreading. These characteristics are related to the property of community structure. In the study of complex networks, a network is said to have *community structure* if the nodes of the network can be easily grouped into sets of nodes such that each set of nodes is densely connected internally, between which connections are sparse.

Communities may thus overlap with each other. For example, people may share the same hobbies in social networks [28], some predator species have the same prey species in food webs [13] and different sciences are connected by their interdisciplinary domain in co-citation networks [20]. However, most of heuristic algorithms

Q. Wang (✉) · E. Fleury
DNET (ENS-Lyon/LIP Laboratoire de l'Informatique du Parallélisme/INRIA Grenoble Rhône-Alpes), Lyon, France
e-mail: qinna.wang@ens-lyon.fr

E. Fleury
e-mail: eric.fleury@inria.fr

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*, Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_2,

are proposed for partition detection, whose results are disjoint communities or partitions. A *partition* is a division of a graph into disjoint communities, such that each node belongs to a unique community. A division of a graph into overlapping (or fuzzy) communities is called a *cover*. We devote this paper to the detection of overlapping community structure.

In order to provide the exhaustive information about overlapping community structure of a graph, we introduce a novel quality function to measure the quality of the overlapping community structure. This quality function is derived from Reichardt and Bornholdt's work [25] and explains the quality of community structure through the energy of spin system.

Moreover, we propose a novel method called fuzzy detection for identifying overlapping nodes and detecting overlapping communities. It applies an existing and very efficient partition detection technique called Louvain algorithm [6]. When running the Louvain algorithm in a graph, we observe that some nodes are grouped together with different community members in distinct partitions. These oscillating nodes are possible overlapping nodes.

This paper is organized as following: we introduce related work in Sect. 2; next, we discuss the modified modularity for covers in Sect. 3; in Sect. 4, we describe our fuzzy detection in details, and applied to networks in Sect. 5 for which the community structure is already known from other studies, our method appears to give excellent agreement with the expected results; in Sect. 6, when applied to networks for which we do not have other information about communities, it gives promising results which may help us to understand better the interplay between network structure and function; finally, we give the conclusion and our future work in Sect. 7.

2 Related Work

2.1 Definition and Notation

Many real world problems (biological, social, web) can be effectively modeled as networks or graphs where nodes represent entities of interest and edges mimic the interactions or relationships among them. A graph $G = (V, E)$ consists of two sets V and E, where $V = \{v_1, v_2, \ldots, v_n\}$ are the nodes (or vertices, or points) of the graph G and $E \subseteq V \times V$ are its links (or edges, or lines). The number of elements in V and E are denoted by n and m, respectively.

In the context of graph theory, an adjacency (or connectivity) matrix $\mathbf{A}$ is often used to describe a graph G. Specifically, the adjacency matrix of a finite graph G on n vertices is the $n \times n$ matrix $\mathbf{A} = [A_{ij}]_{n \times n}$, where an entry A_{ij} of $\mathbf{A}$ is equal to 1 if the link $e_{ij} = (v_i, v_j) \in E$ exists, and zero otherwise.

A *partition* is a division of a graph into disjoint communities, such that each node belongs to a unique community. A division of a graph into overlapping (or fuzzy) communities is called a *cover*. We use $\mathscr{P} = \{\mathscr{C}_1, \ldots, \mathscr{C}_{n_c}\}$ to denote the partition, which is composed of n_c communities. In $\mathscr{P}$, the community to which the node v

belongs to is denoted by σ_v. By definition we have $V = \cup_1^{n_c} C_i$ and $\forall i \neq j$, $C_i \cap C_j = \emptyset$. We denote a cover composed of n_c communities by $\mathscr{S} = \{S_1, \ldots, S_{n_c}\}$. In $\mathscr{S}$, we may find a pair of community S_i and S_j such that $S_i \cap S_j \neq \emptyset$.

Given a community $\mathscr{C} \subseteq V$ of a graph $G = (V, E)$, we define the internal degree k_v^{int} (respectively the external degree k_v^{ext}) of a node $v \in \mathscr{C}$, as the number of edges connecting v to other nodes belonging to $\mathscr{C}$ (respectively to the rest of the graph). If $k_v^{\text{ext}} = 0$, the node v has only neighbors within $\mathscr{C}$: assigning v to the current community $\mathscr{C}$ is likely to be a good choice. If $k_v^{\text{int}} = 0$ instead, the node is disjoint from $\mathscr{C}$ and it should better be assigned to a different community. Classically, we note $k_v = k_v^{\text{int}} + k_v^{\text{ext}}$ the degree of node v. The internal degree k^{int} of $\mathscr{C}$ is the sum of the internal degrees of its nodes. Likewise, the external degree k^{ext} of $\mathscr{C}$ is the sum of the external degrees of its nodes. The total degree $k_{\mathscr{C}}$ is the sum of the degrees of the nodes of $\mathscr{C}$. By definition: $k_{\mathscr{C}} = k_{\mathscr{C}}^{\text{int}} + k_{\mathscr{C}}^{\text{ext}}$.

2.2 Current Work

We then review existing methods for detecting overlapping community structure and discuss the shortcomings of these approaches.

Baumes et al. [4] proposed a density metric for clustering nodes. In their method, nodes are added into clusters if and only if their fusion improves the cluster density. Under this condition, the results really depend on seeds for network clustering. The seed can be a random node or a disjoint community. As shown in their results, there is a huge difference in the number of communities based on different types of seeds.

Lancichinetti et al. has made many efforts in cover detection including fitness-based function [14] and OSLOM (Order Statistics Local Optimization Method) [16]. The former is based on the local optimization of a k-fitness function, whose result is limited by the tunable parameter k, and the later uses the statistical significance [15] of clusters with an expansive computational cost as it sweeps all nodes for each "worst" node. For the optimization, Lancichinetti et al. [16] propose to detect significant communities based on a partition. They detect a community by adding nodes, between which the togetherness is high. This is one of popular techniques for overlapping community detection. There are similar endeavours like greedy clique expansion technique [17] and community strength-based overlapping community detection [29]. However, as they applied Lancichinetti et al. [14]'s k-fitness function, the results are limited by the tunable parameter k.

Some cover detection approaches are based on other basis. For example, Reichardt et al. [25] introduced the energy landscape survey method, and Sales Pardo et al. [26] proposed the modularity-landscape survey method to construct a hierarchical tree. They aim at detecting fuzzy community structure, whose communities consist of nodes having high probability together with each other. As indicated in [26], they are limited by scales of networks.

Evans et al. [7] proposed to construct a *line graph* (a *line graph* is constructed by using nodes to represent edges of the original graphs) which transforms the problem

of node clustering to the link clustering and allows nodes shared by several communities. The main drawback is that, in their results, overlapping communities always exist.

The problem of overlapping community detection remains.

3 Modularity Extensions

Modularity has been employed by a large number of community detection methods. However, it only evaluates the quality of partitions. Here, we first introduce a novel extension for covers, which is combined with the energy model Hamiltonian for the spin system [25]. Second, we review some existing modularity extensions for covers and discuss the cases which these existing extensions may fail to capture. Studies show that our proposed modularity extension is able to avoid their shortcomings.

3.1 A Novel Modularity

Many scientists deal with the problems in the area of computer science based on principles from statistical mechanics or analogies with physical models. When using spin models for clustering of multivariate data, the similarity measures are translated into coupling strengths and either dynamical properties such as spin-spin correlations are measured or energies are interpreted as quality functions. A ferromagnetic Potts model has been applied successfully by Blatt et al. [24]. Bengtsson and Roivainen [5] have used an antiferromagnetic Potts model with the number of clusters as input parameter and the assignment of spins in the ground state of the system defines the clustering solution. These works have motivated Reichardt and Bornholdt [25] to interpret the modularity of the community structure by an energy function of the spin glass with the spin states. The energy of the spin system is equivalent to the quality function of the clustering with the spins states being the community indices.

Let a community structure be represented by a spin configuration $\{\sigma\}$ associated to each node u of a graph G. Each spin state represents a community, and the number of spin states represents the number of communities of the graph. The quality of a community structure can thus be represented through the energy of spin glass. In [25], a function of community structure is proposed, whose expression is written as:

$$\mathscr{H}\big(\{\sigma\}\big) = -\sum_{i \neq j} (A_{ij} - \gamma p_{ij}) \delta(\sigma_i, \sigma_j). \tag{1}$$

This function (Eq. 1) can be written in the following two ways:

$$\mathscr{H}\big(\{\sigma\}\big) = -\sum_{s} \big(m_{ss} - \gamma [m_{ss}]_{p_{ij}}\big) = -\sum_{s} c_s \tag{2}$$

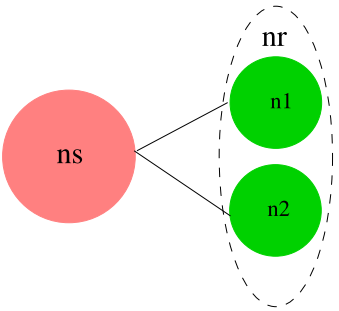


Fig. 1 Example of $[\cdot]_{p_{ij}}$, where the union of clusters n_1 and n_2 is n_r such that $n_1 \cup n_2 = n_r$ and the cluster n_s belongs to the rest of the graph

and

$$\mathscr{H}\big(\{\sigma\}\big) = \sum_{s<r} \big(m_{sr} - \gamma [m_{sr}]_{p_{ij}}\big) = \sum_{s} a_{sr}, \tag{3}$$

where for each community $\mathscr{C}_s$, we note m_{ss} the number of links within $\mathscr{C}_s$, m_{sr} represents the number of links between a community $\mathscr{C}_s$ and another community $\mathscr{C}_r$, $[m_{ss}]_{p_{ij}}$ and $[m_{sr}]_{p_{ij}}$ are the expected number of links given a link distribution p_{ij}. The cohesion of $\mathscr{C}_s$ is noted c_s and a_{sr} represents the adhesion between a community $\mathscr{C}_s$ and another community $\mathscr{C}_r$.

We can assume diverse expressions of $[\cdot]_{p_{ij}}$, which is an expectation under the link distribution p_{ij}. In case of Fig. 1 for disjoint clusters n_1 and n_2, the choice should satisfy the following:

1. when n_s is a cluster belonging to the rest of the graph, $[m_{1s}]_{p_{ij}} + [m_{2s}]_{p_{ij}} = [m_{1+2,s}]_{p_{ij}}$;
2. when n_r is an union cluster composed of n_1 and n_2, $[m_{rr}]_{p_{ij}} = [m_{11}]_{p_{ij}} + [m_{22}]_{p_{ij}} + [m_{12}]_{p_{ij}}$.

Similarly, we give a relation for the cohesion of a community n_3 (the whole graph) and two sub-communities n_1 and n_2 with an empty intersection such as $n_1 \cup n_2 = n_3$ and $n_1 \cap n_2 = \emptyset$ (see Fig. 2(a)). From Eqs. 2 and 3, we can easily prove:

$$c_3 = c_1 + c_2 + a_{12} \tag{4}$$

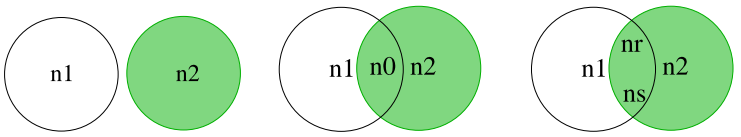
where c_3 denotes the cohesion of n_3 that is the union of n_1 and n_2 with an empty intersection, a_{12} denotes the adhesion between n_1 and n_2, c_1 and c_2 are the cohesions of sub-communities n_1 and n_2 respectively.

Furthermore, we can give the relations for the cohesion of n_3 and two sub-communities n_1 and n_2 in other cases (see Fig. 2).

In the subdivision (see Fig. 2(b)), there is an overlapping cluster n_0 between n_{01} and n_{02}. We write the cohesions for sub-communities n_{01} and n_{02} as:

$$\begin{cases} c^0_{01} = c^0_0 + c_1 + a^0_{01}, \\ c^0_{02} = c^0_0 + c_2 + a^0_{02}, \end{cases}$$

where c^0_{01} and c^0_{02} denote the cohesion of the sub-communities n_{01} and n_{02} respectively, a^0_{01} and a^0_{02} denote the adhesion between n_0 and n_1, n_2. Here, n_0 is shared by n_{01} and n_{02}.



(a) $n_1 \cap n_2 = \varnothing$ (b) $n_{01} \cap n_{02} = n_0$ (c) $n_{rs1} \cap n_{rs2} = n_r \cup n_s$

Fig. 2 Let us denote the union of the clusters n_0 and n_1 by n_{01}. Similarly, we denote the union of the clusters n_0 and n_2 by n_{02}, the union of the clusters n_r and n_s by n_{rs}, the union of the clusters n_1, n_r and n_s by n_{rs1} and the union of the clusters n_2, n_r and n_s by n_{rs2}. Three different subdivisions of the community n_3: (**a**) two disjoint sub-communities n_1, n_2; (**b**) two overlapping sub-communities n_{01}, n_{02} sharing a cluster n_0; and (**c**) two overlapping sub-communities n_{rs1}, n_{rs2} sharing two clusters n_r, n_s, where n_r, n_s are disjoint sub-communities of n_0 such as $n_r \cap n_s = \emptyset$ and $n_r \cup n_s = n_0$

For the adhesion, we have:

$$a^0_{01,02} = a^0_{01} + a^0_{02} + a_{12}$$

between n_{01} and n_{02}.

For the union of $n_3 = n_{01} \cup n_{02}$, we obtain

$$\begin{aligned} c_3 &= c_0 + c_1 + c_2 + a_{01} + a_{02} + a_{12} \\ &= 2c^0_0 + c_1 + c_2 + 2a^0_{01} + 2a^0_{02} + a_{12}. \end{aligned}$$

So we derive

$$c^0_0 = \frac{1}{2}c_0, \qquad a^0_{01} = \frac{1}{2}a_{01} \quad \text{and} \quad a^0_{02} = \frac{1}{2}a_{02}. \tag{5}$$

In the subdivision (see Fig. 2(c)) such as $n_r \cup n_s = n_0$, we replace c_0 and c^0_0 by

$$\begin{cases} c_0 = c_r + c_s + a_{rs}, \\ c^0_0 = c^r_r + c^s_s + a^{rs}_{rs}, \end{cases} \tag{6}$$

where c^r_r and c^s_s denote the cohesion of overlapping sub-communities n_r and n_s respectively. a^{rs}_{rs} denotes the adhesion between overlapping sub-communities n_r and n_s, which satisfies $a^{rs}_{rs} = \frac{1}{2}a_{rs}$ due to Eq. 5.

Therefore, we propose the contribution of a_{rs} for all communities $\{\mathscr{C}_1, \ldots, \mathscr{C}_k\}$:

$$\sum_1^k \frac{1}{|d_r \cup d_s|} a_{rs} = \frac{|d_r \cap d_s|}{|d_r \cup d_s|} a_{rs}, \tag{7}$$

where d_r and d_s denote the community memberships of n_r and n_s, respectively.

The widest used modularity [22] is given by:

$$Q = \frac{1}{2m} \sum_{i \neq j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(\sigma_i, \sigma_j). \tag{8}$$

We rewrite the modularity Q Eq. 8 as:

$$Q = -\frac{1}{m} \mathscr{H}\big(\{\sigma\}\big). \tag{9}$$

Consequently, we can write the quality of an overlapping community structure in the form of the modularity function:

$$Q_{ov} = \frac{1}{2m} \sum_{i \neq j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \frac{|d_i \cap d_j|}{|d_i \cup d_j|}, \tag{10}$$

where d_i and d_j are memberships of nodes i and j, respectively. For a pair of nodes i and j always belonging to the same community such as $d_i \cap d_j = d_i \cup d_j$, their contribution to the modularity is $(A_{ij} - \frac{k_i k_j}{2m})$. For a pair of nodes i and j never belonging to the same community such as $d_i \cap d_j = \emptyset$, their contribution is 0. Otherwise, their contribution is within the range of $[0, (A_{ij} - \frac{k_i k_j}{2m})]$. Furthermore, if the found community structure is a strict partition, its quality Q_{ov} is equal to the initial modularity Q defined by Eq. 8.

3.2 Existing Modularity for Covers

There are other extensions of modularity designed to evaluate the quality of overlapping community structure. However, we are going to prove that they fail to satisfy above necessary constraints.

In the case Fig. 2(c), we assume that n_r is an overlapping node v_i. Similarly for n_s, n_s is an another overlapping node v_j which connects to v_i. The union of v_i and v_j is n_0 such that $n_0 = v_i \cup v_j$. The overlapping communities n_{01} and n_{02} are denoted by $\mathscr{C}_x$ and $\mathscr{C}_y$ of a graph G_{example}, respectively.

Let O_v be the number of communities to which node v belongs. Shen et al. [27] have introduced an extended modularity:

$$Q_{\text{shen}} = \frac{1}{2m} \sum_{i=1}^{n_c} \sum_{v \in \mathscr{C}_i, w \in \mathscr{C}_j, v \neq w} \frac{1}{O_v O_w} \left(A_{vw} - \frac{k_v k_w}{2m} \right) \delta(\sigma_v, \sigma_w). \tag{11}$$

From Eq. 9, it is easy to obtain $a^0_{01_{\text{shen}}}$ derived from Q_{shen} (Eq. 11):

$$a^0_{01_{\text{shen}}} = \frac{1}{2} \sum_{v \in n_0, w \in \mathscr{C}_x \setminus n_0} \left(A_{vw} - \frac{k_v k_w}{2m} \right) + \frac{1}{2} \left(A_{v_i v_j} - \frac{k_{v_i} k_{v_j}}{2m} \right).$$

It fails to satisfy $a^0_{01} = \frac{1}{2} a_0$ (Eq. 5), where

$$a_{01_{\text{shen}}} = \sum_{v \in n_0, w \in \mathscr{C}_x \setminus n_0} \left(A_{vw} - \frac{k_v k_w}{2m} \right) + 2 \left(A_{v_i v_j} - \frac{k_{v_i} k_{v_j}}{2m} \right).$$

In other words, through the definition of Q_{shen}, we obtain different values of the quality in views of Figs. 2(b) and 2(c) although they represent the same cover.

In [21], Tamas Nepusz et al. haved proposed a variant of modularity measure, which is defined by:

$$Q_{\text{fuzzy}} = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) s_{ij}$$

where $s_{ij} = \sum_{k=1}^{n_c} u_{ki} u_{kj}$. The membership degree between node i and community k, u_{ki} satisfies $\sum_{k=1}^{n_c} u_{ik} = 1$.

As we did previously, for node $v_k \in n_0$ in G_{example}, under the assumption: $u_{v_i \mathscr{C}_x} = u_{v_i \mathscr{C}_y} = u_{v_j \mathscr{C}_x} = u_{v_j \mathscr{C}_y} = \frac{1}{2}$, it is easy to obtain

$$s_{v_k v_w} = \begin{cases} 0 & v_w \notin \mathscr{C}_x \cup \mathscr{C}_y, \\ 0.5 & v_w \in \mathscr{C}_x \cup \mathscr{C}_y,\ v_w \notin n_0, \\ 0.25 & v_k \neq v_w. \end{cases} \tag{12}$$

We obtain that

$$a^0_{01_{\text{fuzzy}}} = \frac{1}{2} \sum_{v \in n_0, w \in \mathscr{C}_x \setminus n_0} \left(A_{vw} - \frac{k_v k_w}{2m} \right) + \frac{1}{2} \left(A_{v_i v_j} - \frac{k_{v_i} k_{v_j}}{2m} \right).$$

It also does not satisfy $a^0_{01} = \frac{1}{2} a_0$ (Eq. 5) with $a_{01_{\text{fuzzy}}} = a_{01_{\text{shen}}}$.

By using the novel proposed modified modularity (Eq. 10), we obtain

$$a^0_{01_{\text{ov}}} = \frac{1}{2} \sum_{v \in n_0, w \in \mathscr{C}_x \setminus n_0} \left(A_{vw} - \frac{k_v k_w}{2m} \right) + \left(A_{v_i v_j} - \frac{k_{v_i} k_{v_j}}{2m} \right).$$

It satisfies $a^0_{01} = \frac{1}{2} a_0$ (Eq. 5), therefore we consider that our novel modified modularity is more reasonable to evaluate the quality of overlapping community structure. However, we can not detect covers by optimizing it since overlapping nodes may degenerate the modularity value. For example, in the case Fig. 2(b), the quality can be represented by

$$Q^{\text{cover}}_{ov} = -\frac{1}{m} \mathscr{H}(\{\sigma\}) = -\frac{1}{m} (c_0 + c_1 + c_2 + a^0_{01} + a^0_{02}),$$

where $a^0_{01} = \frac{1}{2} a_{01}$ and $a^0_{02} = \frac{1}{2} a_{02}$. And the quality of the partition is

$$Q^{\text{partition}}_{ov} = \begin{cases} -\frac{1}{m} (c_0 + c_1 + c_2 + a_{01}), & \text{when } \mathscr{P} = \{n_{01}, n_2\}, \\ -\frac{1}{m} (c_0 + c_1 + c_2 + a_{02}), & \text{when } \mathscr{P} = \{n_1, n_{02}\}. \end{cases}$$

We find $Q^{\text{cover}}_{ov} = Q^{\text{partition}}_{ov}$ when $a_{01} = a_{02}$; otherwise, $Q^{\text{cover}}_{ov} < Q^{\text{partition}}_{ov}$ due to $\min(a_{01}, a_{02}) < a^0_{01} + a^0_{02} = \frac{1}{2} a_{01} + \frac{1}{2} a_{02} < \max(a_{01}, a_{02})$. Thus, even in a toy example where clearly there is a clear overlap (see Fig. 2(b)), if the number of links between n_0 and n_1 differs from the number of links between n_0 and n_2 the quality of the cover will be less than the quality of the partition once the difference between the number of links is greater than 0.

To overcome this optimization issue, we propose the method named fuzzy detection not based on modularity like function.

4 Our Method

In this section, we will introduce our method for cover detection named *fuzzy detection*. This novel cover detection heuristic aims at identifying modular overlaps.

Each modular overlap is a group of nodes shared by communities. More precisely, each modular overlap is a possible sub-community shared by several communities. For better understanding, we give two definitions of overlapping nodes: *granular overlaps* and *modular overlaps*. The traditional cover detection methods [4, 14, 16] aims at identifying *granular overlaps*, which are fine grain scale approaches. Each granular overlap is a node connected to distinct communities and it is highly connected to each community. Roughly speaking, a granular overlap is shared by several distinct communities while being intrinsically a member of each of them. As opposed to granular overlaps, modular overlaps imply the hierarchical organization of the graph: each modular overlap is a sub-community shared by several communities.

4.1 Motivation

Our fuzzy detection algorithm is based on the Louvain algorithm [6]. The Louvain algorithm is an efficient partition detection algorithm that provides good partitions with high modularity. It consists of two phases that are iteratively repeated until no more positive gain of modularity is obtained. Initially, all nodes are assigned into a single community. Then, for each node whose move improves the modularity, it will be removed from its current community to the neighbor community which offers the largest gain of modularity. The first phase repeatedly and sequentially sweeps all nodes until no further improvement of modularity can be gained. The second phase builds a new meta graph based on communities found in the first phase. It aggregates nodes of the same community and builds a new network whose nodes are the communities. Once the second phase is completed, the first phase is reapplied to the new network. The two phases are iteratively applied until no more change in community structure or maximum modularity is achieved. In the following, we use iteration to denote the combination of these two phases. The partition found by this algorithm is hierarchical organized, the hierarchy height is determined by the number of iterations. The Louvain algorithm is extremely fast and provides highly optimized partitions with high modularity.

When running several times the Louvain algorithm on the same given network, we observe from a run to another that nodes may be grouped together with different community members in distinct partitions. Since the Louvain algorithm sweeps nodes in a non deterministic fashion (a random permutation of V), it naturally introduces instability which may be a weakness. It turns out that we can take benefit of this instability. By detecting nodes that jump from one community to another between distinct runs, we are in fact able to uncover overlapping nodes. Therefore, we propose a fuzzy detection algorithm which detects groups of nodes having strong probability of appearing in several communities.

4.2 Fuzzy Detection Algorithm

To have the benefit of the potential Louvain algorithm instability [2], we force the algorithm to use a random seed at each run. The random seed makes the nodes be swept in a random permutation during the modularity optimization. Thus, different runs may produce different partitions. By repeating Louvain algorithm, we are able to compute, a co-appearance matrix $\mathbf{P} = [p_{ij}]_{n\times n}$. For each pair of nodes (i, j), p_{ij} of $\mathbf{P}$ represents the probability for the pair nodes i and j appearing in the same community. Having $p_{ij} = 1$ implies that nodes i and j are always in the same community while edges $e = (i, j)$ having a p_{ij} close to 0 implies that edge e connects two different communities. The underlying idea of fuzzy detection approach is thus to detect overlapping communities from a classical partition approach.

Detecting overlapping nodes also allows to detect more stable nodes that always belong together in the same community. In this algorithm, we use the notion of *community cores* to denote communities. Given a community, its *core* is a group of nodes offering high stability against random perturbation. To detect community cores, we're going to remove edges in order to keep only core nodes. First we remove all *external edges*, i.e., all edges $e = (i, j)$, having a connection probability p_{ij} less than a threshold α^*. After this pruning phase, a set of disjoint robust clusters is obtained. A *robust cluster* is a group of nodes connected by edges having in-cluster probability larger than or equal to α^*. Note that a given community may have several robust clusters. We choose the community core corresponding to the robust cluster having the maximum size. The notion of external edges was used in [8] where authors add a random noise over the weight of the edges of the network (equally distributed between $[-\sigma, \sigma]$). Once community cores are identified, we continue iteratively, following the Louvain approach. Similarly, in our method, we replace the robust clusters by supernodes and connect them through the connection between robust clusters. In this case, the weight of the edge between the supernodes is the sum of the weights of the edges between the identified robust clusters. We run again the Louvain algorithm to compute the probability of robust clusters and community cores to appear in the same community. Finally, we add each robust cluster to the community if they have a high community membership degree such as their probability of appearing in the same community is high.

The global algorithm is shown in Algorithm 2. First, (lines 2–9) we compute the co-appearance matrix $\mathbf{P} = [p_{ij}]_{n\times n}$ by running the Louvain algorithm of Algorithm 1 several times with a random seed. The number of runs is determined by the convergence criteria (line 9):

$$\left\|\mathbf{P}^{k+1} - \mathbf{P}^{k}\right\| = \sqrt{\frac{1}{m}\sum_{(i,j)\in E}\left(p_{ij}^{k+1} - p_{ij}^{k}\right)^2} < \varepsilon, \tag{13}$$

where $\mathbf{P}^k$ represents the result after kth run and p_{ij}^k denotes the statistical probability of nodes i and j to belong to the same community after kth runs (line 5) and ε is a small threshold. Figure 3 illustrates the convergence of the norm when running

Algorithm 1 Louvain algorithm

Require: $G = (V, E)$, l^* a level threshold
Ensure: $\mathscr{P}$ a partition
1: $l \leftarrow 0$; $G_0 \leftarrow G$
2: **repeat**
3: $l \leftarrow l + 1$
4: Initialize a partition $\mathscr{P}_l$ of $G_l(V_l, E_l)$
// First phase: Partition update
5: **repeat**
6: Nodes in a random permutation
7: **for all** Nodes: $v \in V_l$ **do**
8: Move from σ_v to one selected $\sigma_{v'}$ (v' is a neighbor of v)
9: **end for**
10: **until** no more change increases modularity
// Second phase: Construct a new meta graph
11: Replace each community by a node
12: Replace connections between a pair of communities by one weighted edge
13: **until** $\mathscr{P}_l$ is not updated or $l = l^*$.
14: Return $\mathscr{P}$ corresponding to the roots of the hierarchical tree.

fuzzy detection algorithm. We observe that $\|\mathbf{P}^{k+1} - \mathbf{P}^k\|$ decreases as the number k of runs increases.

Then, we detect robust clusters $\{c_1, c_2, \ldots, c_s\} = \mathscr{P}_{\text{sc}}$ (lines 10–13). Given a partition $\mathscr{P}_{\text{opt}}$ which has the maximum modularity among all computed partitions obtained during the first phase, the robust clusters are detected by removing all edges having a probability p_{ij} lower that a given threshold α^* (typically $\alpha^* = 0.9$). A simple illustration is given in Fig. 4.

Finally in the second phase, we identify modular overlaps which have high community membership degrees with several communities. Given a community $\mathscr{C}_i \in \mathscr{P}_{\text{opt}}$, its core $\hat{c}_i$ is the robust cluster $c_j \subseteq \mathscr{C}_i$ having the maximum size, such as:

$$\hat{c}_i = \underset{c_j \subseteq \mathscr{C}_i}{\arg\max} |c_j|. \tag{14}$$

We assign each robust cluster c_j to the community $\mathscr{C}_i$ if and only if their community membership degree $p_{c_j,\hat{c}_i}$ is larger than a threshold β^* such as $p_{c_j,\hat{c}_i} \geqslant \beta^*$ (typically $\beta^* = 0.1$). If one robust cluster is assigned to at least two communities, we call it a *modular overlap*.

In cases where a community consists of several robuster clusters of comparable size, one may tune and increase the value of α^* in order to refine the core identification.

Since fuzzy detection is used to identify modular overlaps, which are subcommunities shared by several communities, we restrict the modular overlaps to have a size greater than 3. We can now introduce the notion of *unstable nodes*, which are nodes connecting communities with few links but are observed to have high co-

Algorithm 2 Fuzzy detection

Require: $G = (V, E)$, α^*, β^*
Ensure: $\mathscr{S}$ an overlapping community covering of V
// STEP 1: Detect robust clusters
$\mathbf{P}^0 \leftarrow 0$; $k \leftarrow 0$; $\text{modularity}_{\max} \leftarrow -\infty$
repeat
  $k \leftarrow k + 1$
  $\mathscr{P} \leftarrow$ Run the Louvain algorithm on G
  Update $\mathbf{P}^k$
  if modularity of $\mathscr{P}$ greater than $\text{modularity}_{\max}$ **then**
    Save the partition $\mathscr{P}$ in $\mathscr{P}_{\text{opt}}$ and update $\text{modularity}_{\max}$
  end if
until $\|\mathbf{P}^k - \mathbf{P}^{k-1}\| \leq \epsilon$
$\mathscr{P}_{\text{sc}} = \mathscr{P}_{\text{opt}}$
for all edge $e = (i, j)$ such that $p_{ij} < \alpha^*$ **do**
  Remove the external edge e from $\mathscr{P}_{\text{sc}}$
end for
// STEP 2: Adjust the membership of robust clusters
Require: $G = (V, E)$, $\mathscr{P}_{\text{sc}}$, $\mathscr{S} \leftarrow \mathscr{P}_{\text{opt}}$
for all $\mathscr{C}_i \in \mathscr{P}_{\text{opt}}$ **do**
  Identify community core: $\hat{c}_i = \arg\max_{c_j \subseteq \mathscr{C}_i} |c_j|$
end for
Compute $\mathbf{P}_{c_i, c_j}$
for all $c_j \in \mathscr{P}_{\text{sc}}$ and $c_j \notin \{\hat{c}_1, \ldots, \}$ **do**
  if $p_{c_j, \hat{c}_i} \geq \beta^*$ **then**
    $S_i \leftarrow S_i \cup c_j$
  end if
end for
Return $\mathscr{S}$

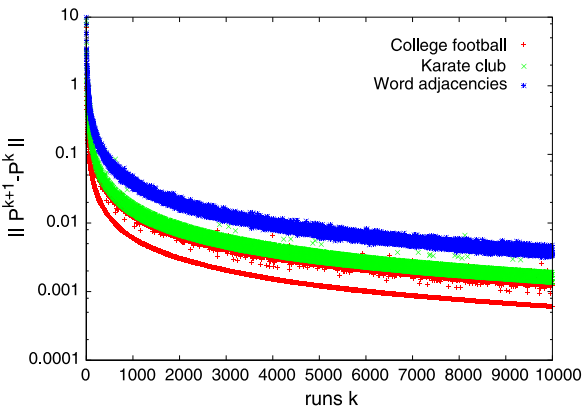


Fig. 3 As the number of runs increases, the shape of the function value Eq. 13 gets closer and closer to 0. The figure shows results on College football [9], Karate club [30] and Word adjacencies [23]

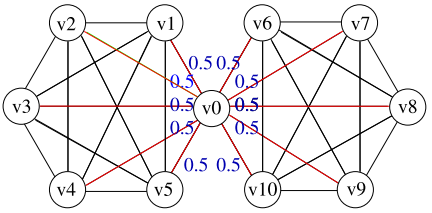


Fig. 4 Illustration of our fuzzy detection on a toy graph which consists of two overlapping cliques. After removing all edges in low probability $p_{ij} = 50$ % (which connect to the node v_0), robust clusters are obtained, concluding $\{v_1, v_2, v_3, v_4, v_5\}$, $\{v_6, v_7, v_8, v_9, v_{10}\}$, and a single v_0

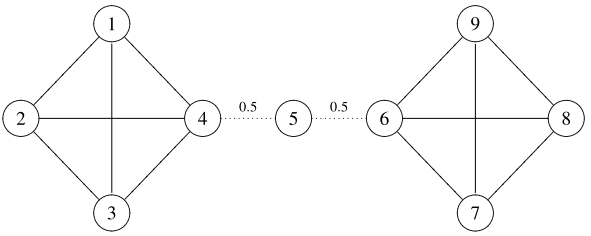


Fig. 5 An example graph that contains a unstable node 5. Node 5 has relatively high membership degrees with two communities ($p = 0.5$). However, it is connected to each community with only 1 link

appearance probability with several communities. Figure 5 illustrates such case. Due to unstable nodes, we only use fuzzy detection to identify modular overlaps.

The running time of fuzzy detection mainly depends on the co-appearance matrix calculation. The complexity to find a partition by the Louvain algorithm is estimated by authors in [6] to be in $\mathcal{O}(m)$, where m is the number of edges in the network (the worst complexity is much higher, but in practice, on real network, Louvain algorithm performs very well). Thus the computational complexity of fuzzy detection is in $\mathcal{O}(Km)$, where K is the number of runs of Louvain algorithm needed before reaching an acceptable convergence of $\mathbf{P}$. Once more, in practice, we take benefit of the efficient Louvain algorithm running time and our fuzzy detection is fast. We experiment storage limitation due to the matrices $\mathbf{P}^k$ and $\mathbf{P}^{k+1}$ more than time computing one.

4.3 Discussion

Our fuzzy detection has applied β^* to determine community memberships. If the threshold β^* increased, the number of modular overlaps decreased; otherwise, more robust clusters are identified as modular overlaps. The criterion we used to fix the optimal β^* value should be based on finding a community structure having the good quality. In the following, we apply our method to a real network and study the modularity by increasing the value of β^*.

Wikipedia is a free encyclopedia written collaboratively by volunteers around the world. A small part of Wikipedia contributors are administrators, who are users with

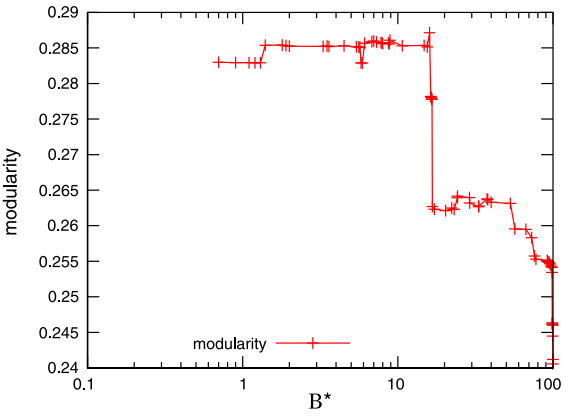


Fig. 6 Performance of fuzzy detection in testing Wikipedia vote network, where the value of the modularity corresponds to the community structure obtained by the relevant β^*. The critical point which corresponds to the maximum modularity is observed

access to additional technical features that aid in maintenance. In order for a user to become an administrator a Request for adminship (RfA) is issued and the Wikipedia community via a public discussion or a vote decides who to promote to adminship. Using the dump of Wikipedia page edit history, 2,794 elections with 103,663 total votes and 7,066 users participating in the elections (either casting a vote or being voted on) are extracted. About half of the votes in the dataset are by existing admins, while the other half comes from ordinary Wikipedia users.[1]

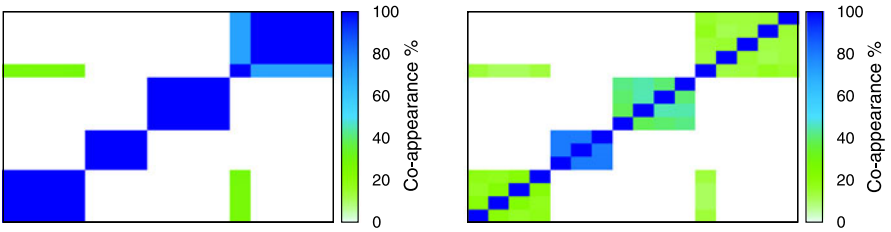
By applying our method to the Wikipedia vote network, we show the modularity by increasing the value of β^*. We observe the critical point: $\beta^* = 18$ % in Fig. 6, which corresponds to the maximum modularity Eq. 10. In practice, we use the value corresponding to the critical point to set β^* which is approximate 10 %. Note that we do not set a high value upon β^* since the obtained membership degree is obtained by modularity optimization. Such that the membership degree $p_{c_j,\hat{c}_i}$ value must be very high if the robust cluster c_j obtains the highest modularity gain with the community $\mathscr{C}_i$ than others. (Even if the modularity gain variance between $\mathscr{C}_i$ and another community is very slight.)

5 Tests of the Method

In the following, we test the performances of fuzzy detection. We have considered a set of synthetic networks and a real network for which the community structure is known. The results show that our fuzzy detection algorithm extracts communities while preserving the *hierarchical organization* and also providing overlaps.

A community structure can be hierarchically ordered when the graph offers several levels of organization/structure at different scales. In this case, the community structure is *hierarchically constructed* by small communities at each level, all nested

[1] http://snap.stanford.edu/data/wiki-Vote.html.



(a) Without l^* (b) $l^* = 1$

Fig. 7 The co-appearance matrix of artificial networks containing hierarchical structure. The *color* corresponds to the probability of nodes in the same community: the *deep color* represents the high probability; the color is *white* if the probability is 0 %

within large communities at higher levels. As an example, one may consider in a social network the granularity of the living place (town), the working place (school) and refine it toward the graduate or class level.

5.1 Synthetic Graphs Containing Hierarchical Structure

First, we apply the fuzzy detection algorithm to an artificial graph containing hierarchical structure [14] and a modular overlap.

The result is shown in Fig. 7. We observe that fuzzy detection extracts communities in hierarchical organization. The graph is composed of 512 nodes, which belong to 16 groups, arranged into 4 supergroups and one group is shared by two supergroups. Every node has an average of $k_1 = 30$ links with nodes in the same micro-community, $k_2 = 13$ links with nodes in the same macro-community but different micro-community. In addition, each node has $k_3 = 5$ links with the rest of the networks. As the modular overlaps has macro-links with two communities, its nodes have a total degree $k = 61$ while the other nodes only have a total degree $k = 48$. This process constructs two hierarchical levels: one consisting of 16 small groups, and the other one composed of 4 supergroups. Figure 7(a) illustrates the co-appearance matrix by running the Louvain algorithm without fixing the level threshold l^* (see Algorithm 1), while Fig. 7(b) provides the result by running the Louvain algorithm with $l^* = 1$. In both figures, the nodes are sorted in the same order corresponding to the robust clusters and the selected partition $\mathscr{P}_{\text{opt}}$. As the distinction among robust clusters is not clear in Fig. 7(a), we use Fig. 7(b) for the visualization. We observe 4 communities and 16 robust clusters, where one robust cluster is shared by two communities. The result agrees with the ground truth.

Remark that, when running our fuzzy detection to identify modular overlaps, we may need to increase the value of α^* to obtain a reasonable community core whose size is larger than the others within the same community. It occurs when one community contains several large robust clusters having comparable size.

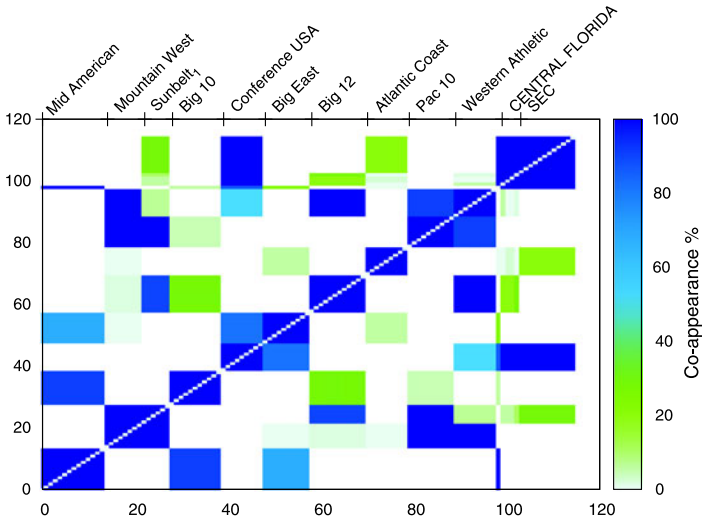


Fig. 8 The co-appearance matrix of college football network by running our fuzzy detection. We order the nodes corresponding to their conferences and mark the conference indices. The *color* corresponds to the probability of nodes in the same community: the *deep color* represents the high probability; the color is *white* if the probability is 0 %

5.2 College Football Network

We also run the fuzzy detection algorithm to real networks. A famous real but small and tractable network is the *US college football* [9] . This network records the schedule of Division I games for the 2000 season: 115 nodes represent teams (identified by their college names) and 613 edges represent regular season games between the two teams they connect. What makes this network interesting [9] is that it incorporates a known community structure. The teams are divided into "conferences" containing around 8 to 12 teams each. Games are more frequent between members of the same conference than between members of different conferences, with teams playing an average of about 7 intra-conference games and 4 inter-conference games fraction of vertices classified correctly in the 2000 season. Inter-conference play is not uniformly distributed; teams that are geographically close to one another but belong to different conferences are more likely to play one another than teams separated by large geographic distances.

In Fig. 8, we illustrate the results: the community "Mountain West Sunbelt" is split into "Mountain West" and "Sunbelt_1", the community "Sunbelt SEC" has a possible subdivision into "Sunbelt_2"[2] and "SEC", and a node "CentralFlorida" is split from the community "Pac 10". Among them, only "Sunbelt_1" is identified

[2]We do not mark "Sunbelt_2" due to the visualization, since its position is too close to "CentralFlorida" in the figure.

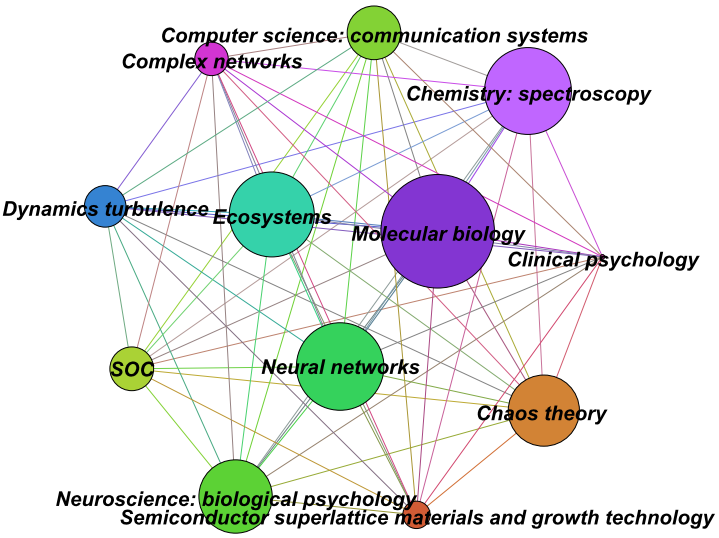


Fig. 9 The community structure of Complex System Science, in which communities are identified by complex systems fields

as a modular overlaps. "CentralFlorida" has high membership degree with different communities, too. But it is a granular overlapping node rather than a modular overlap. In reality, the team "CentralFlorida" did not belong to any conference, and the teams in the "Sunbelt" conference played nearly as many games against Western Athletic teams as they did within their own conference. Therefore, we consider fuzzy detection has a good performance in detecting modular overlaps for this real network.

6 Application to a Real Network: Complex System Science

In this section we consider the application of fuzzy detection to a real network called Complex System Science. It is a co-citation network, whose dataset is composed of articles extracted from the ISI Web of knowledge. Article were published between 2000 and 2009. The network is composed of 141,163 nodes and 19,603,888 links. The nodes correspond to articles containing a set of keywords relevant to the field of complex systems. The weight of the links between articles is calculated through their common references (bibliographic coupling [12]). A link exists between two articles if they share references, meaning that they cite common work which may implies that they are dealing with a same scientific object/domain. More precisely, given two articles (nodes) i and j, each one having a set of references R_i (respectively R_j), there exists a link $e = (i, j)$ between i and j if i and j share at least one reference and the weight is measured by: $w_{ij} = \frac{|R_i \cap R_j|}{\sqrt{|R_i||R_j|}}$.

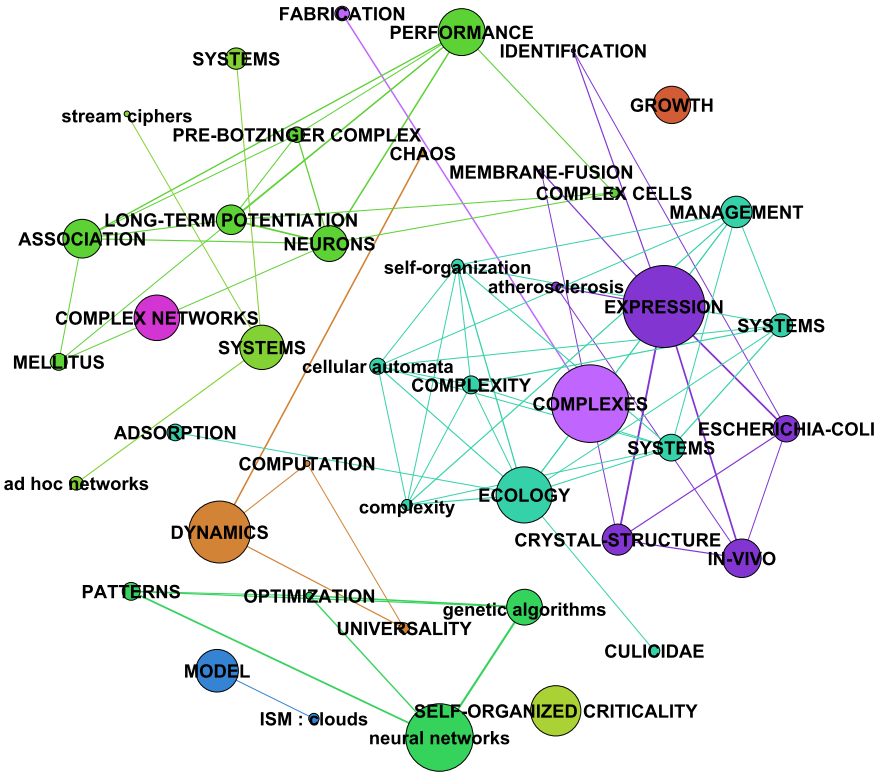


Fig. 10 Results of fuzzy detection on Complex System Science. Robust clusters are marked by the highest frequent topic keywords. Their colors correspond to the relevant communities as shown in Fig. 9

For the visualization, we only show clusters which contain at least 100 nodes.[3] The partition of the graph is shown in Fig. 9. Each community corresponds to a unique color. Our obtained robust clusters are shown in Fig. 10. The color of each robust cluster corresponds to the relevant community in the partition shown in Fig. 9. Only robust clusters belonging to the same community in the partition share the same color.

Figure 9 shows 12 communities (fields or disciplines). Through studies in topic keywords,[4] see Table 1, we observe nearly all important fields of complex systems such as: complex networks, neural networks, self-organization criticality, dynamical systems (chaos theory, dynamics turbulence) and so on [10]. It shows that the community structure of this network reveals the complex systems fields. For more

[3] In [18], the community which has size roughly 100 nodes is good.

[4] We compute the frequency of topic keywords by aggregating the number of units (article), i.e., if only one unite contains the topic keywords "Neurons", the corresponding frequency is 1.

Table 1 Results of communities in the partition. The shown high frequent topic keywords are sorted in descending order and each topic keyword is contained in at least 20 articles

Community	Highest frequent topic keywords	High frequent topic keywords
Neuroscience: Biological Psychology	Brain	Brain, Neurons, Long-Term Potentiation, Association, Expression, Performance, Disease, Model, Synaptic Plasticity, Activation, Complex, Children, Central-Nervous-System, Rat
Chaos Theory	Chaos	Chaos, Dynamics, Systems, Model, Stability, Complexity, Synchronization, Time-Series, Bifurcation, Self-Organization
Chemistry: Spectroscopy	Complexes	Complexes, Self-Organization, Crystal-Structure, Chemistry, Derivatives, Behavior, Films, Polymers, Systems, Phase-Transition, Spectroscopy, Dynamics, Thin-Films, Molecules, Nonlinear-Optical Properties
Complex Networks	Complex Networks	Complex Networks, Dynamics, Small-World Networks, Model, Internet, Evolution, Systems, Organization, Topology, Scale-Free Networks, Metabolic Networks, Web, Graphs
Ecosystems	Ecology	Ecology, Systems, Model, Complexity, Evolution, Dynamics, Management, Growth, Behavior, Self-Organization, Patterns, Simulation, Biodiversity, Models
Molecular Biology	Expression	Expression, Complex, Gene-Expression, Protein, In-Vivo, Activation, Saccharomyces-Cerevisiae, Identification, Gene, Escherichia-Coli, Cells, In-Vitro, Binding, Crystal-Structure, Messenger-Rna, Phosphorylation, Proteins
Semiconductor Superlattice Materials and Growth Technology	Growth	Growth, Gaas, Islands, Molecular-Beam Epitaxy, Self-Organization, Quantum Dots, Surfaces, Films, Photoluminescence, Silicon, Nanostructures, Si(001)
Clinical Psychology	Management	Management, Therapy, Trauma, Experience, Hemorrhage, Surgery, Inhibitors, Optimization, Recombinant Factor Viia, Damage Control, Mortality, Cancer
Neural Networks	Neural Networks	Neural Networks, Model, Systems, Classification, Optimization, Algorithm, Identification, Design, Prediction, Self-Organizing Maps
Soc	Self-Organized Criticality	Self-Organized Criticality, Model, Dynamics, Econophysics, Evolution, Systems, Fluctuations, Behavior, Growth, Turbulence, Noise, Transport, Avalanches, Earthquakes, Patterns, Time-Series
Computer Science: Communication Systems	Systems	Systems, Design, Performance, Channels, Algorithm, Networks, Capacity, Ofdm, Stability, Optimization, Fading Channels, Algorithms, Model, Signals, Codes, Transmission
Dynamics Turbulence	Turbulence	Turbulence, Model, Flow, Simulation, Dynamics, Behavior, Large-Eddy Simulation, Complex Terrain, Plasticity, Flows, Boundary-Layer

Table 2 Results of fuzzy detection: ten high frequent topic keywords contained by modular overlaps between pairs of communities. These high frequent topic keywords are contained in at least 20 articles and are shown in order of descending frequency. The highest frequent topic keywords are shown in bold font

Modular overlaps	High frequent topic keywords	Involving communities
Genetic Association	*Association*, Susceptibility, Polymorphism, Linkage Disequilibrium, Disease, Major Histocompatibility Complex, Linkage, Complex Traits, Risk, Population	Molecular Biology, Neuroscience: Biological Psychology
Discrete-event Systems	*Systems*, Supervisory Control, Petri Nets, Complexity, Discrete-Event Systems, Verification, Design, Automata, Synchronization, Discrete Event Systems	Computer Science: Communication Systems, Ecosystems
Computational Complexity	*Complexity*, Algorithms, Computational Complexity, Algorithm, Networks, Optimization, Time, Systems, Search, Computational-Complexity	Computer Science: Communication Systems, Ecosystems
Astronomy-ISM (Interstellar Medium)	*Turbulence*, Ism: Clouds, Star-Formation, Stars: Formation, Molecular Clouds, Ism: Structure, Ism: Kinematics And Dynamics, Evolution, Radio Lines: Ism, Intergalactic Medium	Dynamics Turbulence, Clinical Psychology
Multi-Agent Systems	*Systems*, Multi-Agent Systems, Multiagent Systems, Design, Agents, Architecture, Multi-Agent System, Framework, Model, Intelligent Agents	Computer Science: Communication Systems, Ecosystems
Visual Cortex	*Complex Cells*, Lateral Geniculate-Nucleus, Cat Striate Cortex, Primary Visual-Cortex, Striate Cortex, Cortical-Neurons, Receptive-Fields, Contrast, Orientation Selectivity, Simple Cells	Neuroscience: Biological Psychology, Neural Networks

details, we analyze robust clusters, which can be considered as sub-communities (subfields or subdisciplines). The result is depicted on Fig. 10, whose description is listed in Table 3. It is no surprise to observe the connection between subfields and fields. For example, the community identified by neuroscience: biology psychology is composed of several clusters, which are also characterized by research topics or theoretical areas. Note that, the study in neuroplasticity supports the treatments of brain damage, long-term potentiation concerns learning and memory, pre-Botzinger complex is essential for respiratory rhythm, and the activities in prefrontal cortex are considered to be orchestration of thoughts and actions in accordance with internal goals. All these subfields refer to the study in neuroscience and biological psychology. It reveals that fuzzy detection extracts communities in hierarchical organization.

In terms of modular overlaps, our results are shown in Table 2. Except astronomy-ISM (Interstellar medium) which acts like a unstable cluster, the rest has a good agreement compared to the reality: discrete-event systems and multi-agents are very common for modeling and analyzing general systems, computational com-

Table 3 Results of fuzzy detection: ten high frequent topic keywords contained by robust clusters. These high frequent topic keywords are contained in at least 20 articles and are shown in order of descending frequency. The highest frequent topic keywords are shown in bold font

Community	Cluster	High frequent topic keywords
Dynamics Turbulence	Flow Over Complex Terrain	*Turbulence*, Model, Flow, Simulation, Complex Terrain, Large-Eddy Simulation, Flows, Behavior, Boundary-Layer, Plasticity
	Astronomy-Ism (Interstellar Medium)	*Turbulence*, Ism: Clouds, Star-Formation, Stars: Formation, Ism: Structure, Molecular Clouds, Ism: Kinematics and Dynamics, Evolution, Radio Lines: Ism, Intergalactic Medium
Computer Science: Communication Systems	Telecommunication System	*Systems*, Performance, Channels, Synchronization, Fading Channels, Capacity, Ofdm, Equalization, Networks, Multiuser Detection
	Control Theory	*Systems*, Stability, Design, Robust Control, Optimization, Linear-Systems, Model-Predictive Control, Stabilization, H-Infinity Control, Model Predictive Control
	Wireless Network	*Ad Hoc Networks*, Sensor Networks, Wireless Sensor Networks, Self-Organization, Networks, Wireless Networks, Clustering
	Cryptography	*Stream Ciphers*, Cryptanalysis, Linear Complexity, Stream Cipher, Sequences
Molecular Biology	Expression	*Expression*, Complex, Gene-Expression, Protein, Saccharomyces-Cerevisiae, Gene, Activation, In-Vivo, Identification, In-Vitro
	Dendritic Cells	*Dendritic Cells*, In-Vivo, Expression, T-Cells, Infection, Complex, Mice, Activation, Major Histocompatibility Complex, Antigen
	Crystal structure of Escherichia Coli	*Crystal-Structure* , Complex, Escherichia-Coli, Binding, Protein, Recognition, Mechanism, Proteins, Molecular-Dynamics, Complexes
	Gene Expression In Escherichia Coli	*Escherichia-Coli*, Gene-Expression, Systems, Expression, Model, Networks, Systems Biology, Protein, Transcription, Rhythms
	Atherosclerosis	*Atherosclerosis*, Inflammation, Expression, Disease, Myocardial-Infarction, In-Vivo, C-Reactive Protein, Smooth-Muscle-Cells, Activation, Low-Density-Lipoprotein

Table 3 (Continued)

Community	Cluster	High frequent topic keywords
Molecular Biology	Membrane Fusion And Exocytosis	*Membrane-Fusion*, Neurotransmitter Release, Exocytosis, Syntaxin, Snare, Complex, Protein, Snare Complex, Transmitter Release
	Proteomics	*Identification*, Proteomics, Mass-Spectrometry, Proteins, Peptides, Protein Identification
Chaos Theory	Chaotic Dynamics	*Chaos*, Dynamics, Systems, Complexity, Stability, Model, Time-Series, Synchronization, Nonlinear Dynamics, Bifurcation
	Quantum Chaos And Universality	*Universality*, Quantum Chaos, Systems, Chaos, States, Model, Random- Matrix Theory, Complex Systems, Fluctuations, Spectra
	Chaos In Population dynamics	*Chaos*, Stability, Dynamics, Population, Permanence, Models, Systems, Bifurcation, Predator-Prey System, Birth Pulses
Neuroscience: Biological Psychology	Neuroplasticity	*RAT*, Neurons, Plasticity, Hippocampus, Brain, Central-Nervous-System, Synaptic Plasticity, Long-Term Potentiation, Food-Intake, Memory
	Long-Term Potentiation	*Long-Term Potentiation*, Synaptic Plasticity, Plasticity, Hippocampus, Nmda Receptor, Glutamate Receptors, Expression, Neurons, In-Vivo, Hippocampal-Neurons
	Genetic Association	*Association*, Susceptibility, Polymorphism, Linkage Disequilibrium, Disease, Major Histocompatibility Complex, Linkage, Complex Traits, Risk, Population
	Pre-Botzinger Complex	*Pre-Botzinger Complex*, In-Vitro, Pre-Botzinger Complex, Brain-Stem, Respiratory Rhythm Generation, Rhythm Generation, Rat, Control of Breathing, Neurons, Pacemaker Neurons
	Prefrontal Cortex	*Performance*, Attention, Fmri, Children, Prefrontal Cortex, Brain, Working-Memory, Cortex, Memory, Activation
	Diabetes Mellitus	*Mellitus*, Glycemic Control, Complications, Hypertension, Randomized Controlled-Trial, Diabetes, Therapy, Risk, Diabetes Mellitus, Management

Table 3 (Continued)

Community	Cluster	High frequent topic keywords
Chemistry: Spectroscopy	Crystal Structure	*Complexes*, Self-Organization, Crystal-Structure, Derivatives, Chemistry, Polymers, Behavior, Films, Nonlinear-Optical Properties, Phase-Transition
	Anodic Alumina	*Fabrication*, Arrays, Films, Anodic Alumina, Anodization, Self-Organization, Growth, Self-Organized Formation, Hexagonal Pore Arrays, Titanium
Soc	Soc	*Self-Organized Criticality*, Model, Dynamics, Econophysics, Evolution, Systems, Fluctuations, Models, Behavior, Turbulence
Ecosystems	Innovation Management	*Management*, Innovation, Economics, Performance, Model, Complexity, Systems, Technology, Firm, Knowledge
	Discrete-Event Systems	*Systems*, Supervisory Control, Petri Nets, Complexity, Discrete-Event Systems, Verification, Design, Automata, Discrete Event Systems, Synchronization
	Computational Complexity	*Complexity*, Algorithms, Computational Complexity, Algorithm, Networks, Optimization, Time, Systems, Search, Computational-Complexity
	Ecosystems	*Ecology*, Dynamics, Evolution, Biodiversity, Patterns, Diversity, Growth, Model, Management, Conservation
	Absorption	*Adsorption*, Sorption, Speciation, Complexation, Humic Substances, Water, Natural-Waters, Kinetics, Ph, Copper
	Cellular Automaton	*Cellular Automata*, Systems, Simulation, Self-Organization, Model, Cellular-Automata, Flow, Cellular-Automaton Model, Traffic Flow, Dynamics
	Multi-agent Systems	*Systems*, Multi-Agent Systems, Multiagent Systems, Design, Agents, Architecture, Multi-Agent System, Framework, Model, Intelligent Agents
	Division of Labor in Insect Societies	*Self-Organization*, Behavior, Division-Of-Labor, Hymenoptera, Ants, Colonies, Formicidae, Social Insects, Swarm Intelligence, Evolution
	Complex Adaptive Systems	*Complexity*, Self-Organization, Chaos, Emergence, Science, Complex Adaptive Systems, Complexity Theory

Table 3 (Continued)

Community	Cluster	High frequent topic keywords
Ecosystems	Malaria	*Malaria*, Culicidae, Identification, Transmission, Complex, Diptera, Africa, Mosquitos, Anopheles-Gambiae Complex, Gambiae Complex
Neural Networks	Neural Networks	*Neural Networks*, Classification, Systems, Model, Self-Organizing Map, Neural Network, Algorithm, Identification, Artificial Neural Networks, Prediction
	Genetic Algorithm	*Optimization*, Genetic Algorithms, Genetic Algorithm, Design, Systems, Neural Networks, Model, Algorithm, Algorithms, Simulation
	Simulated Annealing	*Optimization*, Simulated Annealing, Algorithm, Model
	Gene Expression Patterns	*Patterns*, Self-Organizing Maps, Gene-Expression, Microarray, Identification, Gene Expression, Saccharomyces-Cerevisiae, Cancer, Expression, Classification
Complex Systems	Complex Systems	*Complex Networks*, Dynamics, Small-World Networks, Model, Internet, Networks, Evolution, Scale-Free Networks, Systems, Organization

plexity is a common property of complex systems, and genetic expression [11, 19] studies are often used to determine whether a genetic variant is associated with a disease or trait. Visual cortex is one part of visual systems, which receives visual information for processing images. These results can be validated from the trivial. This also suggests that the interdisciplinarity is important in studies of complex systems.

7 Conclusion

In this paper, we introduce a new extension of modularity for covers and a new method for overlapping community detection. Our definition of modularity is derived from Reichardt and Bornholdt's work [25] and explains the quality of community structure through the energy of spin system. The proposed fuzzy detection benefits from the Louvain algorithm and detects modular overlaps. Modular overlaps are groups of nodes shared by several communities. We have tested our fuzzy detection on synthetic networks and observed its good performances by comparing

to the ground truth. Its application to a real network also hints that our algorithm provides insights in characterizing overlapping nodes.

We hope that our idea and method will provide useful information in the analysis of other types of networks. Possible further applications to dynamic networks will be done for studying effects of overlaps in community changes. We hope to see such applications in the future.

References

1. Albert R, Barabasi A-L (2002) Statistical mechanics of complex networks. Rev Mod Phys 74(1):47–97
2. Aynaud T (2011) Détection de communautés dans les réseaux dynamiques. PhD thesis, Docteur de L'université Pierre et Marie Curie
3. Barmpoutis RM, Murray D (2010) Networks with the smallest average distance and the largest average clustering. arXiv:1007.4031 [q-bio.MN]
4. Baumes J, Goldberg M, Magdon-Ismail M (2005) Efficient identification of overlapping communities. In: Intelligence and security informatics, proceedings, vol 3495, pp 27–36
5. Bengtsson M, Roivainen P (1995) Using the potts glass for solving the clustering problem. Int J Neural Syst 6(2):119–132
6. Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10):P10008. doi:10.1088/1742-5468/2008/10/p10008
7. Evans TS, Lambiotte R (2009) Line graphs, link partitions, and overlapping communities. Phys Rev E 80(1):016105
8. Gfeller D, Chappelier J-C, De Los Rios P (2005) Finding instabilities in the community structure of complex networks. Phys Rev E, Stat Nonlinear Soft Matter Phys 72(5):056135
9. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci USA 99:7821–7826
10. Grauwin S, Beslon G, Fleury E, Franceschelli S, Robardet C, Rouquier J-B, Jensen P (2012) Complex systems science: dreams of universality, interdisciplinarity reality. J Am Soc Inf Sci Technol 63(7):1327–1338
11. Hugot JP, Chamaillard M, Zouali H, Lesage S, Cézard JP, Belaiche J, Almer S, Tysk C, O'Morain CA, Gassull M, Binder V, Finkel Y, Cortot A, Modigliani R, Laurent-Puig P, Gower-Rousseau C, Macry J, Colombel JF, Sahbatou M, Thomas G (2001) Association of nod2 leucine-rich repeat variants with susceptibility to Crohn's disease. Nature 411(6837):599–603
12. Kessler MM (1963) Bibliographic coupling between scientific papers. Am Doc 14(1):10–25
13. Krause AE, Frank KA, Mason DM, Ulanowicz RE, Taylor WW (2003) Compartments revealed in food-web structure. Nature 426(6964):282–285
14. Lancichinetti A, Fortunato S, Kertesz J (2009) Detecting the overlapping and hierarchical community structure in complex networks. New J Phys 11:033015
15. Lancichinetti A, Radicchi F, Ramasco JJ, Fortunato S (2010) Finding statistically significant communities in networks. PLoS ONE 6(4):e18961. doi:10.1371/journal.pone.0018961
16. Lancichinetti A, Radicchi F, Ramasco JJ (2009) Statistical significance of communities in networks. Phys Rev E 81:046110. arXiv:0907.3708 [physics.soc-ph]
17. Lee C, Reid F, McDaid A, Hurley N (2010) Detecting highly overlapping community structure by greedy clique expansion. In: Proceedings of the 4th SNA-KDD workshop. arXiv:1002.1827 [physics.data-an]
18. Leskovec J, Lang KJ, Dasgupta A, Mahoney MW (2009) Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. Internet Math 6(1):29–123. arXiv:0810.1355

19. Limbergen JV, Russell RK, Nimmo ER, Torkvist L, Lees CW, Drummond HE, Smith L, Anderson NH, Gillett PM, McGrogan P, Hassan K, Weaver LT, Bisset WM, Mahdi G, Arnott ID, Sjoqvist U, Lordal M, Farrington SM, Dunlop MG, Wilson DC, Satsangi J (2007) Contribution of the nod1/card4 insertion/deletion polymorphism +32656 to inflammatory bowel disease in northern Europe. Inflamm Bowel Dis 13(7):882–889
20. Michon F, Tummers M (2009) The dynamic interest in topics within the biomedical scientific community. PLoS ONE 4(8):e6544–08
21. Nepusz T, Petroczi A, Negyessy L, Bazso F (2008) Fuzzy communities and the concept of bridgeness in complex networks. Phys Rev E, Stat Nonlinear Soft Matter Phys 77(1):016107
22. Newman MEJ (2004) Analysis of weighted networks. Phys Rev E 70:056131
23. Newman MEJ (2006) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E 74:036104
24. Pu S, Wong J, Turner B, Cho E, Wodak SJ (2009) Up-to-date catalogues of yeast protein complexes. Nucleic Acids Res 37(3):825–831
25. Reichardt J, Bornholdt S (2006) Statistical mechanics of community detection. Phys Rev E 74(1):016110
26. Sales-Pardo M, Guimera R, Moreira A, Amaral L (2007) Extracting the hierarchical organization of complex systems. Proc Natl Acad Sci USA 104(39):15224–15229
27. Shen HW, Cheng XQ, Guo JF (2009) Quantifying and identifying the overlapping community structure in networks. J Stat Mech Theory Exp P07042. doi:10.1088/1742-5468/2009/07/P07042
28. Traud A, Kelsic E, Mucha P, Porter M (2009) Community structure in online collegiate social networks. J Stat Mech Theory Exp 2009:P07042
29. Wang XH, Jiao LC, Wu JS (2009) Adjusting from disjoint to overlapping community detection of complex networks. Phys A, Stat Mech Appl 388(24):5045–5056
30. Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol 1(33):452–473

Constructing and Analyzing Uncertain Social Networks from Unstructured Textual Data

Fredrik Johansson and Pontus Svenson

Abstract Social network analysis and link diagrams are popular tools among intelligence analysts for analyzing and understanding criminal and terrorist organizations. A bottleneck in the use of such techniques is the manual effort needed to create the network to analyze from available source information. We describe how text mining techniques can be used for extraction of named entities and the relations among them, in order to enable automatic construction of networks from unstructured text. Since the text mining techniques used, viz. algorithms for named entity recognition and relation extraction, are not perfect, we also describe a method for incorporating information about uncertainty when constructing the networks and when doing the social network analysis. The presented approach is applied on text documents describing terrorist activities in Indonesia.

Keywords NLP · Relation extraction · SNA · Social network analysis · Text mining

1 Introduction

Visualization of network structures has long been an important tool for police and intelligence analysts who are investigating criminal or terrorist networks (also referred to as *dark networks* [1, 2]). However, social network analysis (SNA) is a rich research field that contains more than just visualization. It has in recent years also become increasingly popular among analysts and investigators to use SNA-related techniques such as node centrality measures and community detection algorithms. An example of how such techniques can be applied is to use them when deciding whom to bring in for questioning in criminal investigations [3]. In [4] it is discussed how SNA metrics can be used to judge which members of terrorist networks that are

F. Johansson (✉) · P. Svenson
Swedish Defence Research Agency (FOI), SE-164 90, Stockholm, Sweden
e-mail: frejoh@foi.se

P. Svenson
e-mail: ponsve@foi.se

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_3,

most important to influence or remove in order to succeed with destabilization of the network. Another example of the use of SNA for intelligence analysis is the capture of Saddam Hussein, which according to [5, 6] was a result of a trace of tribal and family linkages of Hussein, in which certain individuals who may have had close ties to him were identified. The popularity of using SNA for this kind of tasks can be illustrated by the incorporation of SNA functionality in the Analyst's Notebook, which is a tool developed by the company i2, often used by police and intelligence analysts.

A bottleneck with the use of SNA techniques is the actual construction of the social network. In the case of access to structured database information such as phone records, passenger lists, etc., it is rather straightforward to build the network from data (although preprocessing such as record linkage/entity matching [7, 8] often is needed). But how do we proceed when the needed data is not readily available in structured form? Tremendous amounts of useful information can be hidden in the ever-increasing flood of unstructured textual data, but it is usually not possible for human intelligence analysts to read through all available text documents of the phenomenon to be analyzed and manually structure the relevant information in a database. To account for this problem, we suggest the use of natural language processing (NLP) and text mining techniques to automatically extract entities and their relations from massive amounts of unstructured textual data and make use of the extracted information to automatically create social networks. The resulting networks will obviously rarely be of as high quality as if a skilled human analyst read through all the unstructured text and created the network manually. However, as will be argued in this chapter, the networks obtained with the proposed method will be good enough to give a first view of a network to be analyzed, and can easily be improved upon through manual refinement of the results.

Most of the existing work on social networks assume networks with binary relationships, with links having either the value 1 (present) or 0 (absent). However, as observed in [9], social network datasets are often incomplete and prone to observation error, e.g., due to inherent vagueness of human-informant reliability and bias. Since the uncertainties can be expected to be even larger for networks constructed automatically from unstructured text than for manually constructed networks (and since information within the intelligence domain often is unreliable), the problem of incorporating uncertainty can be argued to be even more important in this case. For this reason, we suggest a method for incorporating uncertainty into our automatically created social networks.

The rest of this article is structured as follows. Firstly, a brief presentation of various SNA measures is provided in Sect. 2, illustrated within the context of intelligence analysis. In Sect. 3, research related to managing uncertainty in social networks is presented. Furthermore, a definition of uncertain social networks sufficient for the work presented in this article is given. Next, the problems of named entity recognition and relation extraction from unstructured text are discussed in Sect. 4, together with an overview of related work. In Sect. 5, we suggest a method for constructing uncertain social networks from unstructured textual data, implemented into the tool CACTUS. An experiment for illustrating and evaluating the

suggested method is presented in Sect. 6, together with the acquired results. A discussion is provided in Sect. 7, where we also present ideas for future work. Finally, the main conclusions are summarized in Sect. 8.

2 SNA in the Context of Intelligence Analysis

A social network is often represented as a graph $\mathcal{G} = (V, E)$ consisting of a set of nodes V and a set of edges E, where the nodes (vertices) typically are used to represent actors in the networks (e.g., persons, teams, or organizations), while the edges (sometimes referred to as ties in the terminology relevant to social networks) represent relationships among actors (such as kinship, communication, business relationship, etc.). The edges can be either directed or undirected, depending on the type of network that is modeled. In this example, we will limit the discussion to an undirected social network, where the nodes represent various criminals and where the edges represent relationships among them. Once a graph has been constructed in one way or another (e.g., through interviews, questionnaires, direct observation, data from archival records, or automatic construction from unstructured text), there are a number of *node centrality* measures that can be used to determine the importance of a node in the network. Examples of such node centrality measures are *degree centrality*, *closeness centrality*, and *betweenness centrality* [10]. The degree centrality of a node is simply the number of edges that the node is part of. A strength of this measure is that it is very simple to calculate; only the local structure around the node has to be considered when calculating it. This is, however, also a disadvantage, since it does not take the global structure of the network into account. In contrast, closeness centrality takes such global structure into consideration, as it is defined as the inverse sum of shortest distances to all other nodes from a focal node. In this way, it measures how "close" a node is to all other nodes, saying something about how quickly the node can reach other nodes.[1] Finally, the betweenness centrality measures the proportion of *geodesics* [11], i.e., shortest paths between two nodes in the network, that flow through the particular node. The more shortest paths that pass through a node, the more central to the network it can be argued to be. Removing nodes with maximum betweenness from the network will typically result in large increases in minimum distances among the other nodes [12]. The nodes with high betweenness are often referred to as *information brokers* or *gatekeepers* [11], and are important to identify and remove if the aim is to disrupt the network.

In addition to measuring the impact of individual actors in the network, it can be of importance to analyze properties of the network as a whole. *Centralization* is one such property, measuring the relative difference between the highest and lowest values for the betweenness centrality measure over all nodes in the graph. This measure gives an idea of the variability of centralities among nodes in the network [13].

[1] A downside with closeness centrality is that it is not applicable to networks with several disconnected components. A possible solution for this is to consider the inverse closeness centrality instead.

The most centralized network topology possible is the star topology, so centralization results should be interpreted as the degree of inequality in the network as a percentage of that of a perfect star network of the same size. Another widely used social network analysis concept is that of *density*, defined as the ratio of the number of edges in the graph and the maximum possible number of edges in a graph with the same number of nodes [11, 13]. Hence, the lowest possible density is 0 (no edges present in the graph) and the highest possible density is 1 (a complete/fully connected graph). In general, density is inversely related to the size of the network: the larger the social network, the lower the density [14]. This is due to the fact that the number of possible edges grow very rapidly with the size of the network, while most nodes often have quite few relations.[2]

When analyzing social networks, it becomes obvious that they usually consist of several smaller *communities* (also known as *subgroups* or *clusters*). These communities typically have more and stronger links within the subgroup, than to the nodes outside the subgroup. The nodes connecting two or more communities are often the gatekeepers, as discussed above. To find out how many subgroups a network consists of and to which cluster a certain node belongs is an important problem often referred to as *community detection* or *subgroup identification*. It must be mentioned that the concept of community is not well-defined and depends on what kinds of edges that are considered: depending on the type of relations that are used, different community structures might be found, simply because each person belongs to several different groups (e.g., a person is linked to both work colleagues and old school friends, with different types of links). However, it is nevertheless meaningful to talk about detecting clusters or communities, provided that it is clear that it is a specific kind of community that one is interested in (e.g., one relating scientific authors).

To illustrate the SNA concepts and measures that have been introduced above, we will use the social network shown in Fig. 1. Assume that this network represents known criminals and the known relations among these criminals (in reality, the network is completely made-up). A quick look on this network reveals that the node with highest degree centrality is *Doris* (with a value of 7), closely followed by *Niel* (with a degree centrality of 6). A further analysis of the network with appropriate software (e.g., Gephi [16] or Pajek [17]) reveals that the situation is the same also for closeness and betweenness centrality (i.e., *Doris* in top, closely followed by *Niel*). Hence, it becomes quite obvious that these individuals are worth a closer investigation in case of a planned disruption of the network.

Analyzing the properties of the network as a whole, the density is 0.138, meaning that 13.8 % of all possible edges are present in the network. Likewise, the centralization of the network is 17.4 %. Finally, a community detection algorithm (the Louvain method [18]) has been applied to the social network, resulting in the three communities shown in Fig. 2 (each color represents a community, and the nodes' size represent their degree centrality). Such information can e.g., be used for finding out to which criminal group a specific individual belongs.

[2]Many real-world networks are scale-free, i.e., their number of edges follow a power law distribution [15].

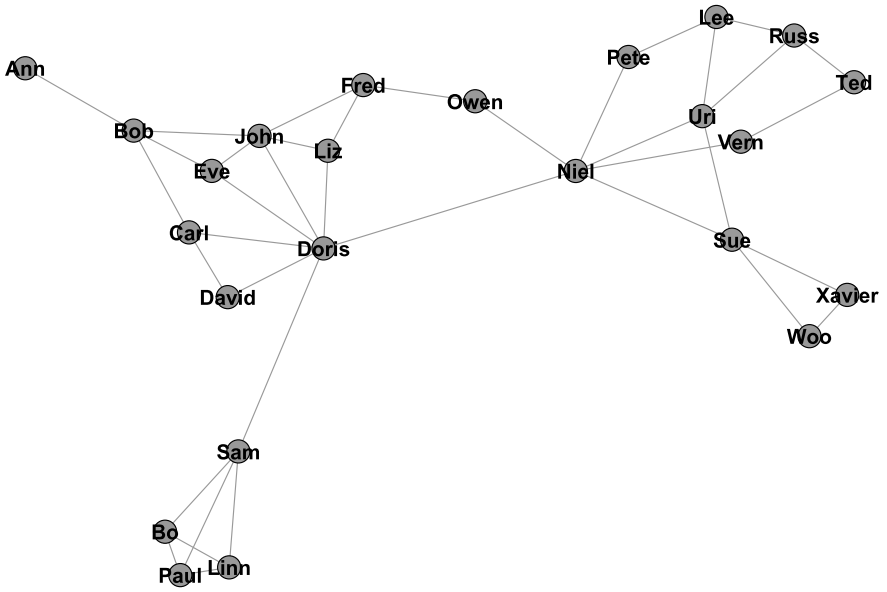


Fig. 1 Example of a social network

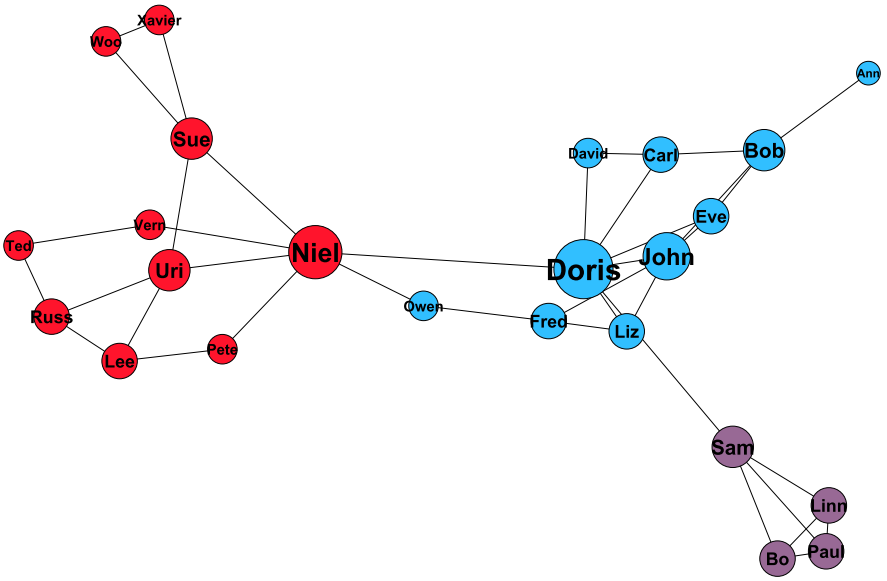


Fig. 2 Communities of the criminal network

3 Uncertain Social Networks

An inherent property of many real-world social networks is that they are associated with varying degrees of uncertainty. This is especially true for large-scale social networks, as highlighted in [19]. For small-scale networks it is in general easier for researchers to be deeply involved in the collection and construction process (e.g., by using interviews and other observational techniques), while one often has to rely on other types of techniques when constructing larger-scale networks (e.g., by using surveys or "snowballing" techniques). The problem with uncertain data becomes evident when constructing social networks automatically from text, due to named entity recognition (NER) and relation extraction techniques being far from perfect, and that not all types of relations will be explicit in unstructured text. However, also smaller-scale social networks are prone to errors and uncertainties due to the inherent unreliability and bias of human informants [9] or factors such as non-response in network surveys [20]. Dark networks and other types of networks of interest to intelligence analysts are particularly vulnerable to uncertainty, due to the imperfect nature of the sources relied upon for building the networks (human intelligence, open source intelligence, signal intelligence, etc.).

A number of experiments have been conducted in recent years, where the robustness of centrality measures under imperfect or missing data has been studied, e.g., [9, 21, 22]. In [22] it is investigated how sensitive various centrality measures are to random errors (missing or extra nodes and edges) in observed networks. Although their study is limited to random error in randomly generated networks, the results suggest that measures of centrality are rather robust to small amounts of error. Similar results are obtained in [9], where it also is discovered that the degree of effect on accuracy is dependent on the actual topology of the network (e.g., random, scale-free, or cellular). The study presented in [21] differs from the others in that random sampling is made based on "real" social networks (i.e., constructed from empirical social studies rather than being artificially generated). Although no strong claims in terms of generalizability of the results are made, the findings indicate that it can be useful to apply centrality measures to networks even though there is some amount of missing data.

Despite the interest of studying which effect "erroneous" edges and nodes will have on the analysis of networks, not much research seems to have been devoted to the incorporation of uncertainty into social networks and SNA, as noted in [23, 24]. An attempt to combine SNA and fuzzy set theory is presented in [25], where fuzzy linguistic expressions such as "strong" or "weak" relationships among nodes are modeled using fuzzy sets. This may be useful for statements such as:

"Anna is a close friend to Lisa"

but does not fit for all kind of situations. In [19] it is argued that probabilistic databases may be useful for managing uncertainty in SNA data. To strengthen this claim an uncertain diffusion network is presented as a motivating example, but very few details are given in their paper. Some approaches to adding uncertainty handling capabilities to network analysis (possibility theory, Dempster-Shafer theory, and

random set theory) are outlined in [23], where it also is suggested that simulation-based methods can be used to analyze networks containing uncertain data. That kind of simulation-based methods are used in [24, 26], where the available (uncertain or imperfect) information is used to create an ensemble of certain networks that is consistent with the original uncertain network using Markov Chain Monte Carlo (MCMC) sampling. Community detection algorithms are applied to each sampled network in the ensemble, whereupon the set of detected community structures are merged into a single community structure for the uncertain network. Additionally, the idea to use Dempster-Shafer theory to handle uncertainty in networks is described more thoroughly in [24]. The main advantage of using Dempster-Shafer theory to represent uncertainty is that it is possible to quantify the degree of ignorance in the problem. The simplest way of understanding Dempster-Shafer theory is to think of it as a generalization of probability theory where the basic concept is called *probability mass* instead of probability, and an event X is described using three numbers instead of two: we talk of the probability mass associated to:

- X,
- its complement $\neg X$, and
- the union of $X \cup \neg X$ (i.e., everything).

The probability mass assigned to the latter represents ignorance and it is required that the three probability masses sum to 1. Compare this to normal probability theory where we deal with the probabilities of X and $\neg X$ which must sum to 1, and where there is no room to express fundamental uncertainty. Dempster-Shafer probability masses for an event can also be represented by intervals giving the *Belief* (i.e., everything that explicitly favours the event) and *Plausibility* (everything that does not explicitly contradict the event). There is a large literature on the interpretation of Dempster-Shafer theory and its relations to probability theory.

Although not much research has been devoted to the incorporation of uncertainty into social networks, it is more common to study and analyze various weighted networks. Usually, the weights take on integer values (e.g., for modeling the number of direct flights between two cities or the number of papers co-authored by two scientists), but a rather intuitive way to model an uncertain network is to instead use weights that are restricted to taking on a value in the unit interval $[0, 1]$ for modeling link uncertainty. This is not general enough for all purposes of uncertainty modeling for social networks since there for example also can be uncertainties regarding the existence of nodes, or relationships in more complex structures such as triads or other larger cliques. Nevertheless, for the purposes of this paper, the expressiveness offered by real-valued weights is enough. Therefore, we here use the following definition of an uncertain network:

Definition 1 Let $\mathcal{G} = (V, E, P)$ be an uncertain network, with V and E denoting the set of nodes and edges, and P representing an adjacency matrix of probabilities associated with edges in the graph, so that $P_{ij} \in [0, 1]$ denotes the probability that there should be an edge E_{ij} between the nodes V_i and V_j.

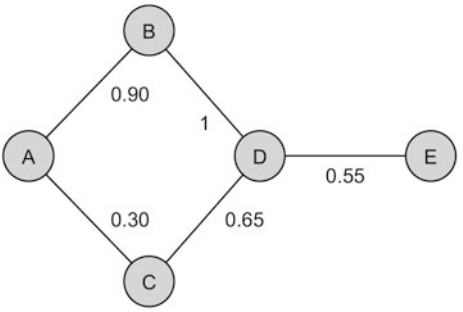


Fig. 3 Example of an uncertain network

An example of such an uncertain network is illustrated in Fig. 3.

An important question then becomes how we can analyze such an uncertain network? Since we are essentially using a type of weighted network for representing our uncertain networks, we can use metrics developed for weighted networks to analyze our uncertain networks. As stated in [27], most social network measures are defined solely for the binary case, and are thus unable to deal with weighted networks directly. However, as observed in [28], weighted networks are in many ways behaving the same as unweighted multigraphs. This observation is the foundation for many of the existing techniques for analysis of weighted graphs. As proposed in [28] and [29], the degree of a node in a weighted network can be defined as the sum of the weights attached to it. This weighted degree is sometimes also referred to as node strength [27]. A direct mapping of this result suggests that we can calculate the degree centrality d_i of a node V_i in an uncertain network $\mathscr{G}$ as:

$$d_i = \sum_j P_{ij}. \tag{1}$$

In the same manner, many other useful generalizations of measures originally developed for unweighted networks have been made for weighted networks, such as eigenvector centrality [28], betweenness centrality [12, 30], and closeness centrality [12]. Since the mappings of these measures from weighted networks to uncertain networks (as we define them in this work) are trivial, we refer the interested readers to the above mentioned papers for presentations of the weighted node centrality measures. Recently, other generalizations of weighted versions of various node centrality measures have been suggested in [27], allowing for taking the number of ties (degree centrality) or the number of intermediary nodes (betweenness and closeness centrality) into consideration by the inclusion of a tuning parameter α.

In addition to applying node centrality measures to uncertain networks, it can also be of interest to detect community structures in the network. As a motivating example, this is of importance for this work since we would like to be able to discriminate between people actually being part of dark networks, and other people co-occurring in texts by pure chance or for other legitimate reasons. A community detection algorithm for weighted networks has been suggested in [28], which is a generalization of the Girvan-Newman algorithm [31]. First, the betweenness centrality of the graph is calculated (ignoring the weights). After that, the betweenness

is divided by the weight of the corresponding edge, whereupon the edge with the highest resulting score is removed. Then the betweenness for the new graph is calculated, and so on, until a suitable number of communities have been created (the best number of communities can be decided using the well-known *modularity* measure [32]). For community detection, however, it is not as straightforward to apply the algorithms for weighted networks to uncertain networks. The reason for this has to do with the effects of correlations between the uncertain edges in a network, that will lead to simple algorithms based on weighted networks giving the wrong results, see [26].

4 Extraction of Entities and Relations from Unstructured Text

In order to be able to create uncertain networks from unstructured text, it is necessary to first be able to extract relevant entities and relations from the text. The problem of extracting named entities from unstructured text is briefly presented in Sect. 4.1, and the problem of identifying relations between extracted entities is discussed in Sect. 4.2. A few attempts in literature to create social networks from extracted information are presented in Sect. 4.3.

4.1 Extraction of Named Entities

Named entity recognition (NER) involves the task of processing text automatically in order to discover mentions of proper nouns such as persons, organizations, and locations. In the seventh Message Understanding Conference (MUC-7), the following types of named entities were defined [33]:

- People names, geographic locations, and organizations.
- Dates and times.
- Monetary amounts and percentages.

According to the survey on NER presented in [34], the very first research articles on NER were published in the early 1990s. Early attempts for extracting named entities relied on heuristics and hand-crafted rules, but more recent approaches to NER are mostly based on statistical models, trained using supervised or semi-supervised learning algorithms. Examples of supervised learning techniques that have been used for NER are hidden Markov models (HMMs), decision trees, support vector machines (SVMs), and conditional random fields (CRFs) [34].

Many modern algorithms for NER obtain a high accuracy, where the best are able to get as high as 95 percent precision and recall when trained on domain-specific data [33]. However, these results greatly depend on the choice of domain, and the results can be expected to be significantly lower if no suitable domain-specific training data are available.

4.2 Extraction of Relations

According to [35], the concept of relation extraction was first introduced in the sixth Message Understanding Conference (MUC-6). More lately, it has also been promoted by the Automatic Content Extraction (ACE) program [36]. Formally, the goal of relation extraction can be specified as: "the task of recognizing the assertion of a particular relationship between two or more entities in text." [37]. Most approaches to relation extraction (sometimes also referred to as relation discovery) are based on supervised machine learning algorithms searching for a small set of prespecified semantic relations, learned from manually labeled training examples (see e.g., [38]). Since this kind of approaches require very many training examples, various bootstrapping algorithms have been suggested, reducing the number of training examples needed. All these kinds of extraction can be referred to as targeted (or traditional) relation extraction [39]. This traditional sense of relation extraction will throughout this work be referred to as TRE. In 2007, open relation extraction (ORE), sometimes also referred to as open information extraction, was introduced in [40]. A system for ORE extracts relations without requiring any relation-specific human input, making it more suited for large corporas such as text harvested from the Web [37]. For this kind of problem, the use of conditional random fields [37] and Markov logic networks [41] have been suggested.

Some researchers are also making a distinction between *explicit* and *implicit* relation extraction, where explicit relationships are explicitly stated in text, while implicit relations should be evident from reading a text but are not necessarily explicitly stated [42].

An approach to relation discovery in large text corpora is presented in [35]. Their idea is to extract named entities and collect all instances of entity pairs occurring within a certain distance of each other, as well as the context words intervening the named entities. In order to calculate the context similarities of a pair of entities, a bag of words is formed from all intervening words from all co-occurrences of the two entities, weighing the words in the vector using the popular tf-idf (term frequency-inverse document frequency) measure and applying cosine similarity between the resulting context vectors. This is used to cluster the extracted entity pairs, whereupon the clusters are labeled with the most frequent common words. These common words are also used for labeling the relations between the entities in that cluster.

4.3 Generating Social Networks

Once named entities and relations have been extracted one way or another, we want to create a social network based on the extracted information. The idea to extract social networks from textual data is not new. A technique referred to as "melting pot" is tested in [43], where it is compared to a more traditional (manual) snowballing approach. In their "melting pot" approach, names are extracted from news pages using a tool called DynaLink. For all pairs of actors identified at the same news page, a link is created between the actors. This is not unexpectedly creat-

ing a large network with a large amount of extraneous links, since it is common that people are talked about at a web page without having any tight relationships among each other. In [44], social networks are created from text documents such as personal home pages and publications. The used method builds upon the use of a vector space model where latent semantic indexing (LSI) is used to find similarities between people. This approach is however not suitable for the kind of text documents of interest here. In [45], social networks from 19th century British novels and serials are constructed, where the nodes in the network are extracted characters, and where the edges in between them are created when bilateral conversations between the characters are identified. Hence, their suggested approach relies on components finding instances of quoted speech, identifying conversation among certain characters. Conversation among characters is a useful way to identify this very specific type of relation, but is not useful for finding more general relationships among entities in text. Mesquita et al. [39] extract information networks from blog posts using their system SONEX. Their approach is similar to ours in many aspects, with the main difference that we incorporate uncertainty into our relations and the constructed social network, as further described in Sect. 5.

5 Suggested Approach for Creating Uncertain Social Networks from Unstructured Text

We want to be able to automatically create social networks by extracting named entities and relations from various unstructured text documents. This is indeed a hard problem, associated with many subproblems. First of all, approaches for extracting named entities from text are not perfect, resulting in that named entities present in a text can be missed, or misclassification of non-entities as entities. Secondly, there are very many ways to express various relations in unstructured text. Even though the number of target relations of interest are restricted to one or a few, the extracted information will not be perfect. Moreover, we do not want to restrict ourselves to a few predetermined target relations, and hence, are here dealing with open relation extraction (ORE). Once relations and named entities have been extracted, it should be possible to fuse the results from several texts, and to generate a social network that can be presented to the analyst (who can analyze it further using SNA). The overall architecture for a system fulfilling such a process is shown in Fig. 4.

5.1 Module for Extraction of Named Entities and Uncertain Relations

In our implemented prototype system CACTUS (Creative Analysis and Construction Tool for Uncertain Social networks), we have made use of Python's Natural Language Toolkit (NLTK) for the extraction of named entities and relations, although many other alternative toolkits and packages for natural language processing are available, open source as well as commercial.

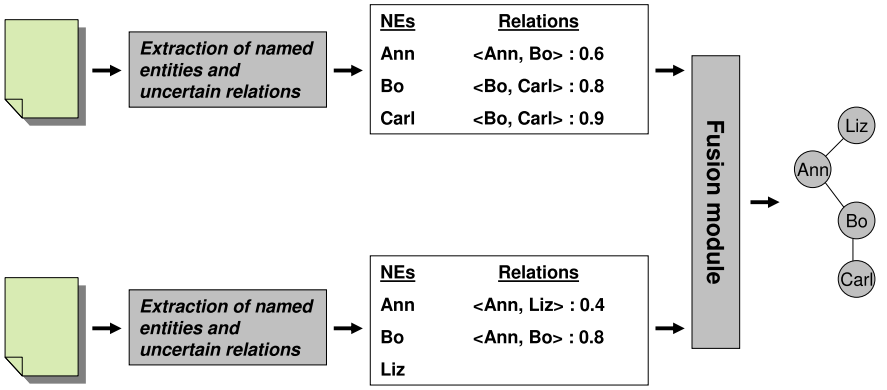


Fig. 4 Illustration of the overall process for generation of social networks from unstructured text

In the used method, we take one or more text document(s) as input. Each text document is segmented into sentences, which then are tokenized through lexical analysis. The resulting tokens are subject to part-of-speech (POS) tagging and noun phrase (NP) chunking, which in its turn is used to extract the named entities from the text. The extracted named entities are classified into the categories PERSON, ORGANIZATION, and GPE, i.e., geo-political entity.[3] Our implemented algorithm goes through each sentence and checks whether it contains any named entities. A list (hashmap) containing all the named entities discovered so far keeps track of how many times each named entity have been discovered in the text. In case a discovered named entity already is in the list, the value is incremented by one, otherwise the entity is added to the list.

Once the named entities in a text document have been identified, next part is to extract the relations between entities. Our algorithm goes through the sentences one by one, searching for pairs of extracted named entities within the individual sentences. If such a pair is discovered and the number of words in between them is less than a specified threshold β, we add the pair to a relation list in a similar fashion to the list of named entities described above. The argument for this is that entities that are far apart are unlikely to be semantically associated [39, 47]. Moreover, we store the strength of the relation. As a heuristic, we judge named entities that are occurring close to each other as more likely to be related than entities with many words in between them. For this reason we suggest using a ramp shaped membership function (see Fig. 5) that weights the strength of the extracted relation between two entities A and B as:

$$s(x) = \begin{cases} 1 & \text{if } x \leq \alpha, \\ 0 & \text{if } x \geq \beta, \\ 1 - (\frac{x-\alpha}{\beta-\alpha}) & \text{otherwise,} \end{cases} \tag{2}$$

[3] A more complete description of the workings of the NER in NLTK can be found in [46].

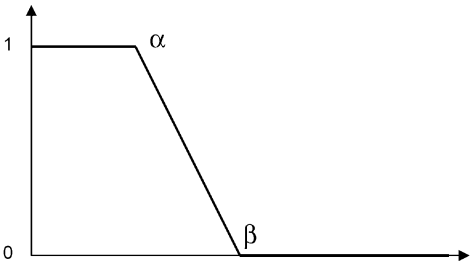


Fig. 5 Used function to weight the strength of extracted relations

where x is the number of words in between the two named entities, α and β are thresholds specified by the user, and where $\beta > \alpha$ (in case of the user wanting to have $\alpha = \beta$, we suggest replacing the suggested ramp function with a step function taking on the value 1 if $x < \beta$, and 0 otherwise).

5.2 The Fusion Module

Once the recognition of named entities and relations is done, next step is to fuse the results from the extraction processes. Such fusion is needed internally within a single document, and in case of several texts, also across several documents. One part of the fusion process is entity matching. This is needed since a unique entity can be referred to in many different ways in unstructured texts. When the entity matching is completed, we have a list of (hopefully) unique entities present in the document(s). Since the entity recognition is likely to generate a number of false positives, we delete the named entities that after matching occurs less than τ times (also deleting the relations of which the named entity is a part). As can be seen in Fig. 4, there can be several extracted strengths for the relation between a pair of entities (both within a single text document and among several text documents). Hence, we need to aggregate these results in order to get a single weight for each relation, before constructing the resulting uncertain social network. There are several alternative ways to calculate the final weights, and there is probably not a single solution that works best for all situations. One way is to simply take the average (or min or max) of the estimated strengths of a relation as the resulting weight. A potential downside with this is that the relation of a pair of entities A and B that occur together in a single sentence is considered to be as certain as the relation of a pair of entities C and D occurring together hundred times, given the same number of words in between the entities. Since this does not make sense in general, an alternative can be to also take the relation's relative frequency into account, i.e., the number of extracted relations of the particular type divided by the total number of extracted relations. In our implementation we have instead chosen to use Dempster's rule of combination (see [48]) for fusing the relation strengths into a single weight for each relation, since this allows for a behavior where several evidences of the same strength (mass) result in a higher weight than a single piece of evidence with the same strength.

In order to apply Dempster's rule, we need to convert the strengths of the extracted relations into so-called probability mass functions. A probability mass function m is a function from the subsets 2^{Θ} of some set Θ to the interval [0, 1] such that $m(\emptyset) = 0$ and $\sum_{x \subseteq \Theta} m(x) = 1$. In our case, Θ is very simple: $\Theta = \{\textbf{Link}, \neg\textbf{Link},\}$, corresponding to the link either being present or not. Recall that we are now considering the problem of fusing all information regarding the relation between two fixed entities (A and B above), and it is hence possible to use such a simple representation. We define a probability mass function m_i for each extracted strength to be fused and let $m_i(\{\textbf{Link}\}) =$ the strength s determined using the ramp function described in Fig. 5, and $m_i(\Theta) = 1 - m_i(\{\textbf{Link}\})$. This corresponds to the assumption that the strength of an extracted relation is the belief that there is a link present. The reason for using a Dempster-Shafer representation is simply that we cannot take the complement to this strength to represent evidence against there being such a link—instead it simply represents uncertainty, which the Dempster-Shafer representation allows us to represent using $m_i(\Theta)$.

Fusion of mass functions is done using Dempster's rule:

$$m_{1,2}(A) = \frac{1}{1-K} \sum_{x \cap y = A} m_1(x) m_2(y), \tag{3}$$

where $K = \sum_{x \cap y = \emptyset} m_1(x) m_2(y)$ is a normalization constant.

For the simple case where $\Theta = \{\textbf{Link}, , \neg\textbf{Link},\}$ and $m_i(\{\neg\textbf{Link},\}) = 0$ (i.e., we do not have any evidence that explicitly talks against there being links) we can simplify the calculations and simply compute

$$m_{1,2}(\Theta) = m_1(\Theta) m_2(\Theta), \tag{4}$$

and then use the normalization requirement

$$m\big(\{\textbf{Link}\}\big) + m\big(\{\neg\textbf{Link}\}\big) + m(\Theta) = 1 \tag{5}$$

to determine $m(\{\textbf{Link}\})$ for each link. Note that this computation is only valid in the special case where $m_i(\{\neg\textbf{Link}\}) = 0$ for all i!

The result from the fusion module is a social network consisting of the extracted entities and their weighted relationships. The results are stored as a .graphML-file, making it easy to load it into a suitable SNA-tool.

6 Experiment

To illustrate the potential use of the developed method for intelligence analysis, we have tested it on a set of documents describing terrorist activities in Indonesia. The documents are open reports downloaded from the non-governmental organization International Crisis Group (ICG). The downloaded files are originally in pdf-format, but have been converted to txt-format in order to be processable in our implemented system.

In our experiment, we have used the parameter settings: $\alpha = 3$, $\beta = 10$, and $\tau = 5$, where the latter has been chosen so as to minimize the number of non-entities being classified as entities and at the same time avoid deleting important entities.

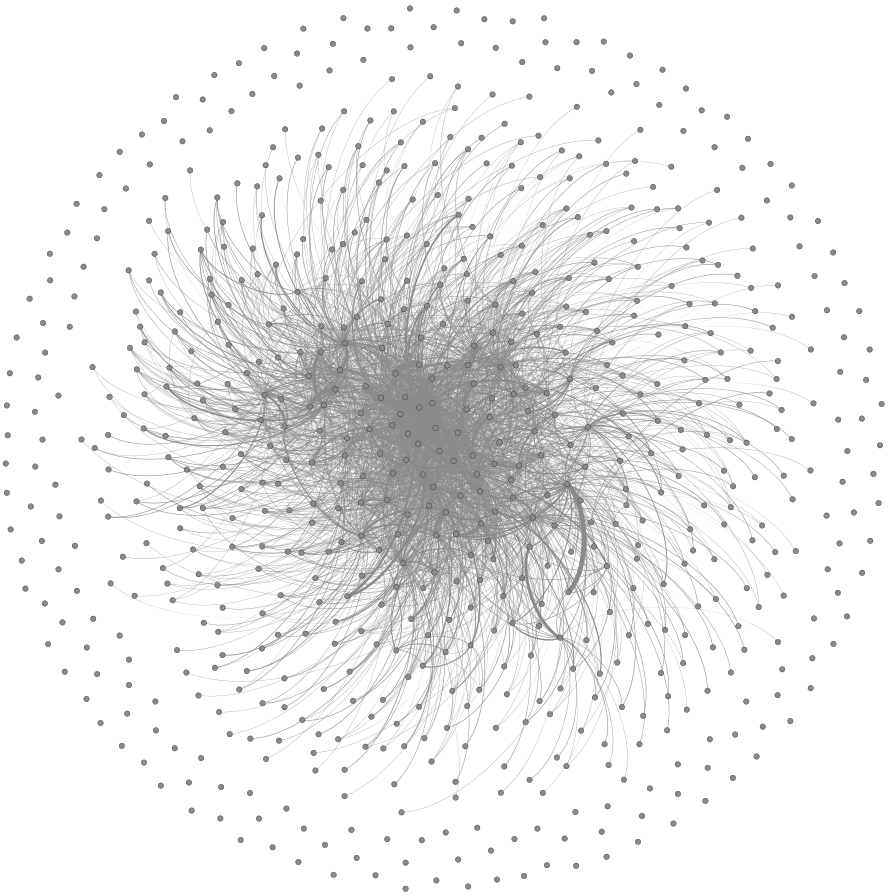


Fig. 6 Illustration of the complete social network obtained in the experiment

When applying these parameter settings to one of the downloaded reports,[4] we have after the fusion process extracted 709 entities, and a total of 3141 relationships from the unstructured text report. In Fig. 6, an overview of the full network is shown. As can be seen, the full network is cluttered and quite hard to derive useful information from, despite filtering out all entities occurring less than five times. However, by using an interactive program such as Gephi, it becomes easy to zoom into interesting parts of the network. As an example, we have from the large network in Fig. 6 selected a smaller part of the obtained social network (the ego network of the former terrorist leader Abdul Aziz). This ego network consists of 57 nodes and 252 edges and is shown in Fig. 7. The analyst can derive much useful information from such an ego network that has been constructed automatically from unstructured text. As

[4] http://www.crisisgroup.org/en/regions/asia/south-east-asia/indonesia/043-indonesia-backgrounder-how-the-jemaah-islamiyah-terrorist-network-operates.aspx

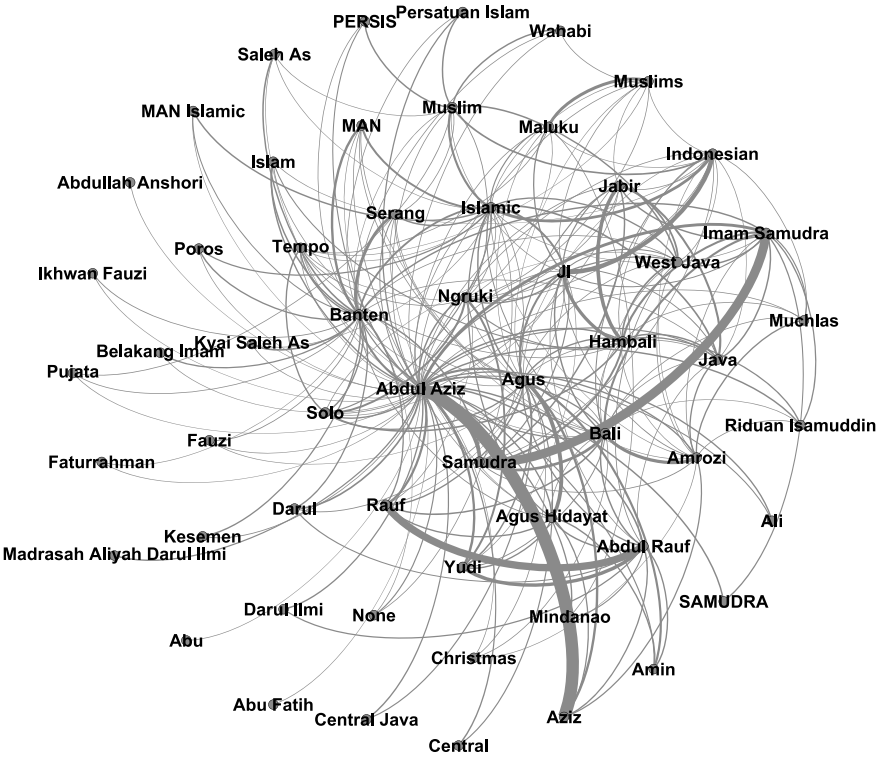


Fig. 7 Illustration of the ego network for Abdul Aziz extracted from the larger Jemaah Islamiyah terrorist network

an example, a quick view of the ego network reveals to an analyst that the entity Abdul Aziz is closely related to Imam Samudra (since no entity matching was made prior to the construction of the network, Samudra, SAMUDRA, and Imam Samudra are treated as separate entities) and Abdul Rauf, and is related to e.g., West Java and Bali. In reality, Abdul Aziz, also known as Imam Samudra, was an Indonesian who was a recognized member of Jemaah Islamiyah and who was sentenced to death for organizing the Bali bombings in 2002, where Abdul Rauf was one of his associates.

Once the network has been constructed, it is also straightforward for the analyst to apply various SNA metrics of interest in order to learn more about the underlying social network. In Fig. 8, we have chosen to filter out the nodes with highest weighted node degree centrality (see Eq. 1). This gives the analyst a good idea of which entities that are most central in the text document. Analyzing the created social network, we can see that one of the most central nodes is Hambali, who is the former military leader of the Jemaah Islamiyah. Another central node is Darul Islam, which is another Indonesian Islamist group. A less successful result is the inclusion of ICG, which is the organization that has published the text document used as input to the experiment. Such problems can, however, easily be corrected by

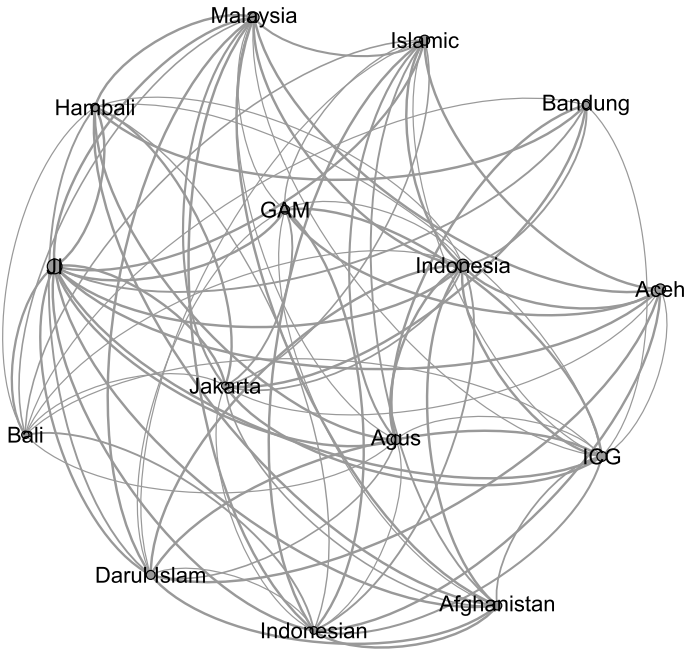


Fig. 8 Social network for the nodes with highest degree centrality

the analyst through manual refinement of the acquired social network (by removing or combining nodes).

7 Discussion

The proposed method should be taken as a first attempt to take uncertainty into account when constructing social networks automatically from unstructured text, not as the final solution to the problem. As demonstrated in Sect. 6, the current version of the system can be of great value for a human analyst, but there is still room for improvements. A potential limitation with the current approach is that relations can only be extracted on a sentence-level, i.e., they must occur between entities within a single sentence. Connected to this problem is the lack of coreference resolution in the suggested method. These limitations hinder us from detecting relations such as the following:

"Carl is very tall. He is the brother of Liz."

To incorporate methods for coreference resolution is therefore a natural next step.

In the current approach, we have not taken the type of relationship among entities into account. As future work, it could be of interest to also add likely labels to the created networks, e.g., by using a clustering method similar to the one sug-

gested in [35]. If a label of the extracted relations can be obtained, this would allow for a better understanding of the created social network. However, an edge between two entities can potentially consist of more than one type of relation, making it far from trivial to extract suitable labels for the relations. The relation extraction methodology as such can also be improved upon in future work. Interesting statistical methods able to perform traditional TRE as well as ORE have been suggested in recent literature (see e.g., [41]), so one potential road forward for improving the implemented system's performance could be to combine such a method with the ideas that have been presented in this article. Another interesting area to investigate further is the choice of function for estimating the relation strength. To assume that words close to each other are likely to have some kind of relation is not unusual in existing literature, but the idea that the strength should be linearly decreasing with the number of words in between is not well-founded, and should rather be taken as a heuristic that can be improved later on.

In order to say something more specific about how good the proposed method is, a more quantitative evaluation would be beneficial. However, as stated in [44], there exists no objective method for evaluating extracted social networks. The constructed social networks are, however, a result of the entity and relation extraction processes, so one way to partially evaluate the overall system could be to evaluate the quality of the extracted entities and relations on a MUC or ACE benchmark data set. A potential problem with this is that such datasets are extracted from news stories, which is not necessarily representative for more general text corpora. It is also common to manually evaluate the obtained relation extraction results on a few manually selected and annotated sentences. However, as noticed by [39], such a manual evaluation does not scale and easily becomes biased by choosing relations that the system is likely to identify correctly.

Further research is needed on the interpretation of uncertainty in social networks. Here we have used Dempster-Shafer theory since it allows us to represent both the belief that there is a link and the uncertainty using probability mass functions. We emphasize again the difference to standard probability assignments: Dempster-Shafer theory allows us three parameters representing explicit evidence for and against an event being true and ignorance, whereas probability requires us to consider all evidence not explicitly for the event to contradict it. In both cases there is a normalization requirement that reduces the number of parameters. We believe that the Dempster-Shafer representation is very natural for relation extraction, since we can get evidence for there being a link from analyzing, e.g., a web page or document, but the absence of a link in the analyzed documents can not normally be taken as evidence against the link. However, a statement such as "*Anna does not know Alan*" can of course be used as evidence against the link between Anna and Alan, which we in Dempster-Shafer theory represent by assigning some probability mass to $m(\{\neg\mathbf{Link}\})$.

We have in this chapter only touched upon the subject of uncertainty representation in social networks. For instance, we only considered one link at a time and constructed probability mass functions separately for each link. Dempster-Shafer theory allows us to also consider the entire network at the same time. In these cases, it is

similar to a random set formulation of the uncertainty in the network [23, 24, 26]. We plan to look into this in more detail in future work.

8 Conclusions

In conclusion, we have in this work presented a method for automatic construction of social networks from unstructured text documents. In the suggested method, the social network is created by extracting entities and relations from the text. This is accomplished through the use of natural language processing techniques, where named entity recognition is used for the entity extraction, and where the occurrence of pairs of entities in a sentence is used as a simple form of open relation extraction. Since natural language processing techniques are not perfect (and since information within text documents typically are not totally reliable), our proposed method allows for the creation of uncertain relations, through the incorporation of weighted edges. To create such uncertain social networks from unstructured text is a novel research contribution. We used Dempster-Shafer probability mass functions to represent the evidence for links. This allowed us to represent also ignorance and to fuse evidence for a link between two entities A and B from several sources.

To illustrate and test the proposed method, we have implemented it in a tool called CACTUS and applied it on an unclassified text document describing the Indonesian terrorist organization Jemaah Islamiyah. The full network created with the proposed method is quite large (consisting of more than 700 nodes and 3000 relations), but simple filters can easily be applied to look into interesting subparts of the network. By applying a filter for showing only the nodes with highest weighted degree centrality, useful information regarding key players and key locations of the network can be acquired. Likewise, filters can be applied to show ego networks for interesting nodes in the network.

There is still room for improvement of the method, e.g., by also allowing for extraction of more targeted relations from the text in addition to the basic open relation extraction. Nevertheless, the method can already today be of great value for extracting useful information from sources which the intelligence analysts have no time for reading. Obviously, the method can also be applied outside the intelligence domain. More advanced models for representing the uncertainty and for fusing uncertain information could also be used.

Acknowledgements This work was supported by the R&D programme of the Swedish Armed Forces. We would like to express our thanks to the other members of the FOI Information Fusion and Data Mining group and the VIA project for fruitful discussions and valuable feedback.

References

1. Raab J, Milward HB (2003) Dark networks as problems. J Public Adm Res Theory 13:413–439

2. Svenson P, Svensson P, Tullberg H (2006) Social network analysis and information fusion for anti-terrorism. In: Proceedings of the conference on civil and military readiness 2006
3. Zhu B, Watts S, Chen H (2010) Visualizing social network concepts. Decis Support Syst 49:151–161
4. Geffre JL, Deckro RF, Knighton SA (2009) Determining critical members of layered operational terrorist networks. J Defense Model Simul, Appl Methodol Technol 6:97–109
5. Hougham V (2005) Sociological skills used in the capture of Saddam Hussein. http://www.asanet.org/footnotes/julyaugust05/fn3.html
6. Koelle D, Pfautz J, Farry M, Cox Z, Catto G, Campolongo J (2006) Applications of Bayesian belief networks in social network analysis. In: Proceedings of the 4th Bayesian modeling applications workshop during the 22nd annual conference on uncertainty in artificial intelligence
7. Fellegi IP, Sunter AB (1969) A theory for record linkage. J Am Stat Assoc 64(328):1183–1210
8. Dahlin J (2011) Entity matching. Swedish Defence Research Agency, Tech Rep
9. Frantz TL, Cataldo M, Carley KM (2009) Robustness of centrality measures under uncertainty: examining the role of network topology. Comput Math Organ Theory 303–328
10. Freeman LC (1979) Centrality in social networks: conceptual clarification. Soc Netw 1(3):215–239
11. Scott J (2000) Social network analysis, 2nd edn. Sage, Thousand Oaks
12. Newman MEJ (2001) Scientific collaboration networks. ii. Shortest paths, weighted networks, and centrality. Phys Rev E 64:016132
13. Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, Cambridge
14. de Nooy W, Mrvar A, Batagelj V (2005) Exploratory social network analysis with Pajek. Structural analysis in the social sciences. Cambridge University Press, Cambridge
15. Barabási A-L, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
16. Bastian M, Heymann S, Jacomy M (2009) Gephi: an open source software for exploring and manipulating networks. In: Adar E, Hurst M, Finin T, Glance NS, Nicolov N, Tseng BL (eds) Proceedings of the 3rd international AAAI conference on weblogs and social media
17. Batagelj V, Mrvar A (2002) Pajek—analysis and visualization of large networks. In: Mutzel P, Jünger M, Leipert S (eds) Graph drawing. Lecture Notes in Computer Science, vol 2265. Springer, Berlin, pp 8–11
18. Blondel V, Guillaume J, Lambiotte R, Mech E (2008) Fast unfolding of communities in large networks. J Stat Mech, Theory Exp P10008
19. Adar E, Ré C (2007) Managing uncertainty in social networks. IEEE Data Eng Bull 30(2):23–31
20. Kossinets G (2006) Effects of missing data in social networks. Soc Netw 28:247–268
21. Costenbader E, Valente TW (2003) The stability of centrality measures when networks are sampled. Soc Netw 25:283–307
22. Borgatti SP, Carley KM, Krackhardt D (2004) On the robustness of centrality measures under conditions of imperfect data. Soc Netw 28(2):124–136
23. Svenson P (2008) Social network analysis of uncertain networks. In: Proceedings of the 2nd Skövde workshop on information fusion topics
24. Dahlin J, Svenson P (2011) A method for community detection in uncertain networks. In: Proceedings of the European intelligence and security informatics conference, EISIC 2011
25. Yager RR (2008) Intelligent social network analysis using granular computing. Int J Intell Syst 23:1196–1219
26. Dahlin J (2011) Community detection in imperfect networks. Master's thesis, Umeå University
27. Opsahl T, Agneessens F, Skvoretz J (2010) Node centrality in weighted networks: generalizing degree and shortest paths. Soc Netw 32(3):245–251
28. Newman MEJ (2004) Analysis of weighted networks. Phys Rev E 70:056131

29. Barrat A, Barthelemy M, Pastor-Satorras R, Vespignani A (2004) The architecture of complex weighted networks. Proc Natl Acad Sci (PNAS) 101:3747
30. Brandes U (2001) A faster algorithm for betweenness centrality. J Math Sociol 25:163–177
31. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci (PNAS) 99(12):7821–7826
32. Newman MEJ (2006) Modularity and community structure in networks. Proc Natl Acad Sci USA 103(23):8577–8582
33. Feldman R, Sanger J (2007) The text mining handbook—advanced approaches in analyzing unstructured data. Cambridge University Press, Cambridge
34. Nadeau D, Sekine S (2007) A survey of named entity recognition and classification. Linguist Investig 30(1):3–26
35. Hasegawa T, Sekine S, Grishman R (2004) Discovering relations among named entities from large corpora. In: Proceedings of the 42nd annual meeting on association for computational linguistics
36. Doddington G, Mitchell A, Przybock M, Ramshaw L, Strassel S, Weischedel R (2004) The automatic content extraction (ACE) program: tasks, data, and evaluation. In: Proceedings of LREC'04
37. Banko M, Etzioni O (2008) The tradeoffs between open and traditional relation extraction. In: Proceedings of ACL-08: HLT, pp 28–36
38. Zelenko D, Aone C, Richardella A (2003) Kernel methods for relation extraction. J Mach Learn Res 3:1083–1106
39. Mesquita F, Merhav Y, Barbosa D (2010) Extracting information networks from the blogosphere: state-of-the-art and challenges. In: Proceedings of the fourth international conference on weblogs and social media
40. Banko M, Cafarella MJ, Soderl S, Broadhead M, Etzioni O (2007) Open information extraction from the web. In: Proceedings of the 20th international joint conference on artificial intelligence, pp 2670–2676
41. Zhu J, Nie Z, Liu X, Zhang B, Wen J-R (2009) Statsnowball: a statistical approach to extracting entity relationships. In: Proceedings of the 18th international conference on world wide web, ser. WWW '09, pp 101–110
42. GuoDong Z, Jian S, Jie Z, Min Z (2005) Exploring various knowledge in relation extraction. In: Proceedings of the 43rd annual meeting on association for computational linguistics, pp 427–434
43. Morris JF, Anthony K, Kennedy KT, Deckro RF (2011) Extraction distractions: a comparison of social network model construction methods. In: Proceedings of the 2011 European intelligence and security informatics conference, EISIC2011
44. Makrehchi M, Kamel MS (2005) Building social networks from web documents: a text mining approach. In: Proceedings of the 2nd LORNET scientific conference
45. Elson DK, Dames N, McKeown KR (2010) Extracting social networks from literary fiction. In: Proceedings of the 48th annual meeting of the association for computational linguistics, pp 138–147
46. Bird S, Klein E, Loper E (2009) Natural language processing with python: analyzing text with the natural language toolkit. O'Reilly Media
47. Fang Y, Chang KC-C (2011) Searching patterns for relation extraction over the Web: rediscovering the pattern-relation duality. In: Proceedings of the fourth ACM international conference on Web search and data mining, ser. WSDM '11, pp 825–834
48. Shafer G (1976) A mathematical theory of evidence. Princeton University Press, Princeton

Privacy Breach Analysis in Social Networks

Frank Nagle

Abstract This chapter addresses various aspects of analyzing privacy breaches in social networks. We first review literature that defines three types of privacy breaches in social networks: interactive, active, and passive. We then survey the various network anonymization schemes that have been constructed to address these privacy breaches. After exploring these breaches and anonymization schemes, we evaluate a measure for determining the level of anonymity inherent in a network graph based on its topological structure. Finally, we close by emphasizing the difficulty of anonymizing social network data while maintaining usability for research purposes and offering areas for future work.

Keywords Social network · Privacy · Breach · Security · Anonymity

1 Introduction

Over the past 5–10 years, online social networks have rapidly expanded, and as of March 2012 the largest online social network, Facebook, had over 901 million active members.[1] The wealth of information users post in their social network profiles, as well as the underlying structure of the network itself, are appealing datasets for researchers and alluring targets for criminals. Due to the attractiveness of these networks to both groups, researchers have recently begun to study methods for releasing this data in a manner that maintains the privacy of the individuals in the network. This chapter presents an overview of recent work in this field.

The data users disclose in their profiles is wide ranging, and includes everything from email addresses and birthdays to political interests and physical address. Two case studies of the information users reveal [1, 23] found that approximately 90 % of users list a partial or full birthday as well as an email address. These pieces of

[1] Facebook Newsroom Key Facts, http://newsroom.fb.com/content/default.aspx?NewsAreaId=22, accessed May 06, 2012.

F. Nagle (✉)
Harvard Business School, Wyss 100, Soldiers Field, Boston, MA 02163, USA
e-mail: fnagle@hbs.edu

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_4,

information can be used by an attacker to steal the user's identity. To allow the use of such networks for academic research, social network graph owners often utilize a technique known as naïve anonymization where all identifying information is removed from the graph and only a network graph of nodes and edges is released. While this technique helps to prevent the release of private information, below we detail a number of privacy breaches that can still occur with such network graphs. To address these privacy breaches, a variety of new anonymization techniques have arisen. However, these anonymization techniques require a tradeoff of the accuracy of the representation of the original network for anonymity. This balance between anonymity and usability of social network graph data can be difficult to maintain and is an important aspect of the effectiveness of an anonymization technique. In the following sections we explore the various privacy breaches that can occur in online social networks, as well as a number of anonymization techniques that can help to prevent such breaches. Finally, we discuss a method for analyzing the level of anonymity inherent in a network graph, and consider areas for future work.

1.1 Graph Notation

To enable a complete discussion of social network graphs, we introduce a few graph notation concepts. For a more complete description of social network graph analysis, we refer the reader to [29]. A graph G, can be represented as a set of vertices, V, and a set of edges, E, where

$$G = (V, E). \tag{1}$$

We assume edges in G are undirected. The set V is a list of all of the vertices or nodes within the graph,

$$V = \{v_1, v_2, \ldots, v_n\}. \tag{2}$$

For our usage, we will not allow self edges. Therefore, the set E is a list of all of the edges or connections within the graph,

$$E = \left\{e_{ij} = (v_i, v_j) \mid v_i \in V, \text{ and } v_j \in V, \text{ and } i \neq j\right\}. \tag{3}$$

A sub-graph contains a subset of V and the relevant edges from E, formally,

$$H = \left\{\left(V', E'\right) \mid V' \subseteq V,\ E' \subseteq E, \text{ and } \left((v_i, v_j) \in V' \rightarrow e_{ij} \in E'\right)\right\}. \tag{4}$$

For the purpose of analyzing social network graphs, we can represent user i in the social network as a vertex v_i. All social networks have the concept of a *relationship* between users, whether it be "friendship" on Facebook, "following" or "followers" on Twitter, or "connections" on LinkedIn. These relationships are treated as edges. For example, if user v_1 and user v_2 are friends on Facebook, then the edge $e_{1,2}$ would exist within E. Directed social network graphs can be converted into undirected graphs by removing the directionality of the relationship and stating that $e_{i,j}$ implies $e_{j,i}$. Our work in this chapter will focus on undirected graphs. Figure 1

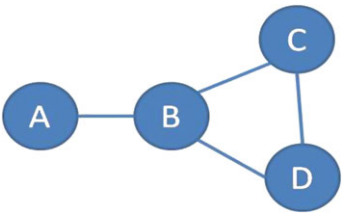


Fig. 1 A sample network graph

shows a sample network graph with vertices, represented by circles, connected via edges, represented by lines. The neighborhood of a vertex is the set of vertices it is connected to. More formally,

$$N(v_i) = \{v_j \mid e_{ij} \in E,\ v_i \neq v_j\}. \tag{5}$$

In Fig. 1, the neighborhood of vertex C is vertices B and D. The degree of a vertex is the size of the neighborhood. We define degree as follows,

$$degree(v_i) = \left|N(v_i)\right|. \tag{6}$$

In Fig. 1, the degree of vertex A is 1, C and D is 2, and B is 3. The clustering coefficient of a vertex shows how densely connected the vertex's neighbors are. We define a clustering coefficient CC_i of the node i as follows:

$$CC_i = \frac{2S}{degree(v_i) * (degree(v_i) - 1)}, \tag{7}$$

where $degree(v_i) \geq 2$, and $S = |\{e_{jk} \mid \forall v_j,\ v_k \in N(v_i),\ e_{jk} \in E\}|$.

The clustering coefficient of a node will be 1 if all of its neighbors are connected to each other, and 0 if none of its neighbors are connected to each other. In Fig. 1, the clustering coefficient of B is 0.333, while the clustering coefficients of C and D are 1. Because A only has one neighbor, it's clustering coefficient is non-existent.

2 Privacy Breaches in Social Networks

At a high level, a privacy breach occurs when personal information is disclosed and used in a manner that the owner did not intend, potentially leading to abuse. For example, when a person provides personal and medical history to a doctor's office, if that information is not protected properly and it is revealed to other patients then, according to our definition, a privacy breach has occurred. Even a person's Internet search history can be considered personal data, the aggregation and exposure of which is considered a privacy breach. Such a breach occurred when the Internet-service provider (ISP) America On-Line (AOL) disclosed 20 million searches by 650,000 users over a three month period. The person from AOL who disclosed the information had intended it be used by academics to research the search patterns of AOL customers. However, the data was not anonymized in such a way that would prevent individual users from being identifiable by their search histories. The only anonymization used was to replace individual usernames with a unique number.

Although this prevented the direct association of a person's search results with their name, researchers were able to aggregate successive searches by the same person to gain insight into who that person was, and in some cases, they were able to identify the person in the real world. The most famously identified person from this data was Thelma Arnold, who agreed to let the New York Times interview her and discuss how they discovered who she was [6]. While this breach is not specifically related to social networks, it does show that even if it data is anonymized, the connections between various data points can often reveal sensitive information.

Social network privacy breaches can be broken into three types of privacy breaches: interactive, active, and passive. These three levels reflect the degree of interaction an attacker needs to have with the network to successfully exploit the breach. An *interactive privacy breach* requires the attacker to create an account in a social network and to use that account to directly interact with the accounts of his targets. Such a breach allows the attacker to gain access to potentially sensitive information within a user's profile. An *active privacy breach* requires an attacker to create an account in a social network and to structure his connections so that his account is easily recognizable when the social network graph is naïvely anonymized and released for research purposes. This type of breach allows an attacker to quickly identify their location in the naïvely anonymized social network graph and to then use their knowledge of their location in the real world network to de-anonymize other nodes in the graph. Finally, a *passive privacy breach* allows an attacker to identify sensitive information about a social network graph without requiring the attacker to have an account on the social network. We discuss various manifestations of these three breaches in the following sections.

An adversary attempting to cause one of the above breaches may have differing levels of knowledge about the network. The least effective adversary has no knowledge whatsoever of the network and is known as a *zero-knowledge adversary*. This adversary starts with no knowledge of the underlying network structure and needs to determine the network structure to cause any of the breaches. When considering most privacy breaches, it is assumed that the adversary at least has access to a naïvely anonymized version of the network, and that is how they will conduct their attack. We call such an adversary a *structural-knowledge adversary*. Here the adversary knows the underlying structure of the network, but does not know any identifiers of the nodes in the network. When a network dataset is naïvely anonymized and released to the public, anyone who downloads the dataset has the knowledge of a structural-knowledge adversary. This is the level of knowledge we consider an attacker to have to conduct passive privacy breaches. In some cases, an adversary may have additional information about the individual they are targeting, such as information about the degree of that individual, or information about the connectivity of the individual's neighbors. We call an adversary with such knowledge a *specific-knowledge adversary*. For our usage, we consider all specific-knowledge adversaries as a subset of structural-knowledge adversaries. Specific-knowledge adversaries are capable of conducting active and targeted passive breaches. Conducting an interactive privacy breach often gives an attacker the ability to have this specific-knowledge when they conduct an active or targeted passive breach.

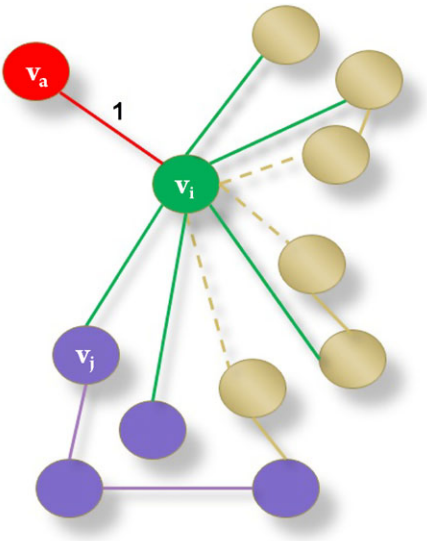
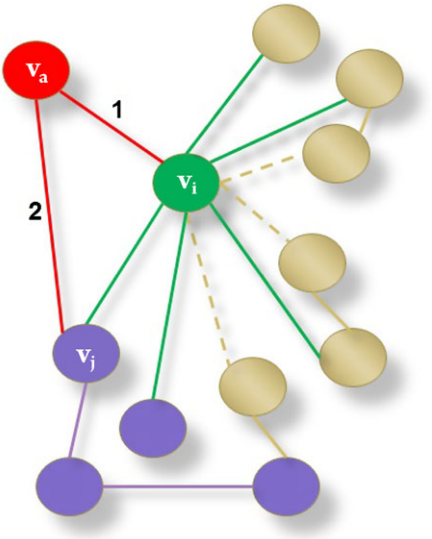


Fig. 2 A friendship privacy breach

2.1 Interactive Privacy Breaches

Attackers who utilize interactive privacy breaches are attempting to gain knowledge of sensitive information about users, potentially for the purpose of identity theft. Of the three breach types, these particular breaches allow an attacker the most amount of information about the target. They also require the attacker to interact with the target directly and to trick the target into disclosing sensitive information to the attacker. Nagle and Singh [23] use a case study with Facebook users to detail two such breaches, a *friendship privacy breach*, and a *trust privacy breach*. When attempting to exploit a friendship privacy breach, an attacker first initiates a request for a friendship, link, following, or other type of connection request with the target. If the target accepts the request, thereby giving the attacker access to sensitive information they would not have otherwise had, then the friendship privacy breach is successful. In Fig. 2, if the attacker, v_a, requests a connection with the victim, v_i, and the victim accepts the request, then a friendship privacy breach has occurred. In their case study, Nagle and Singh found that they were able to successfully conduct a friendship privacy breach with 955 of 5063 target users, or about 19 % of their sample population. The second breach Nagle and Singh discuss, a trust privacy breach, occurs when the attacker is able to exploit the trusted relationship users have with their connections in social networks. When attempting to exploit a trust privacy breach, an attacker initiates a request for a linkage with the connections of someone they are already connected with. Therefore, when the new victim receives the request, they are told that they have a mutual connection with the attacker. If the new victim accepts the request, thereby giving the attacker access to sensitive information they would not have otherwise had, due to the trusted relationship with this mutual connection, then the trust privacy breach is successful. In Fig. 3, if the attacker, v_a,

Fig. 3 A trust privacy breach



already has a connection with v_i, and then requests a connection with a second victim, v_j, who accepts the request, then a trust privacy breach has occurred. In their case study, Nagle and Singh found that they were able to successfully conduct a trust privacy breach with 1963 of 3549 target users, or about 55 % of their second sample population. After establishing the existence of these privacy breaches, Nagle and Singh summarize the potentially sensitive information their attacker profiles were able to obtain.

2.2 Active Privacy Breaches

Active privacy breaches are attempts to de-anonymize nodes in a naïvely anonymized social network graph. They rely on the attacker actively creating an account in the social network before it is anonymized. The attacker structures his connections so that his account is easily recognizable when the social network graph is naïvely anonymized and released for research purposes. The *injection attack* [4] is a type of active privacy breach that requires an attacker to create a known distinct subgraph structure within the graph before it is naïvely anonymized. If this subgraph is unique, then the attacker will be able to identify it within the anonymized data. They also present an alternative type of injection attack that allows an attacker to rely on forming a coalition of neighbors that fashion a semi-unique subgraph. By obtaining this semi-unique subgraph, the coalition can greatly narrow the possible locations of their subgraph within the naïvely anonymized graph, and the list of possible subgraphs diminishes with each new neighbor added to the coalition. Backstrom, Dwork, and Kleinberg [4] found that there are so many distinctly possible subgraph formations in social networks that they could inject as small as a

7-node subgraph into the 4.4 million node LiveJournal blogging social network, and successfully identify their injected subgraph over 90 % of the time.

2.3 Passive Privacy Breaches

Passive privacy breaches have become the most studied of the three breaches due to their direct impact on the ability of social networks to release anonymized graphs to researchers for study. A passive privacy breach allows an attacker to identify sensitive information about a social network graph without requiring the attacker to have an account on the social network. Singh and Zhan [28] define a *node identity breach* as having occurred if an attacker can successfully identify the label, usually a person's name or unique identifier, of a node in a graph that has been anonymized. This type of breach, often considered the most damaging breach, is the main breach considered by many of the other studies in this survey. Additionally, Singh and Zhan define an *edge inference* breach which occurs if an attacker is able to determine that an edge exists between two specific nodes. This type of breach is commonly considered less damaging, but still reveals information about the network and was the focus of inquiry for Zheleva and Getoor [32]. In this study, they found that an adversary that is not part of a social network can ascertain information about private user profiles by viewing friendship links and group membership of public profiles. In a case study of users on Facebook, Zheleva and Getoor found that by analyzing friendship links, they were able to determine the political preference attribute of a private user with 61.8 % accuracy. Additionally, by analyzing group membership, they were able to determine the gender of a private user with 73.4 % accuracy. Their earlier work [31] introduced the concept of *link re-identification*, which occurs when it is possible to infer specific sensitive relationships between nodes in a naïvely anonymized network graph.

Zhou and Pei [33] identified unique subgraphs, similar to those discussed by Backstrom, Dwork, and Kleinberg [4] that occur naturally in social network graphs and therefore allow for *neighborhood attacks* that are similar to the injection attacks above to be carried out in a passive manner. Neighborhood attacks allow for an attacker to identify unique subgraphs in anonymized social network graphs which can lead to node and edge inference breaches. Singh and Schramm [27] identify a more efficient way for storing and searching for these unique subgraphs, that allows for an increased effectiveness of a neighborhood attack.

While all of the breaches discussed above focus on privacy breaches in individual social networks, it is possible for privacy breaches to occur due to the similarities of network structures across multiple social network graphs. Acquisiti and Gross [2] focus on the ability to combine data from numerous social networks in an attempt to re-identify individuals who have accounts on multiple sites. They show that aggregating an individual's personal information from multiple social networks can allow for the estimation of personal identifying information (PII), such as a social security number. Additionally, Narayana and Shmatikov [25] discovered that neighborhood privacy breaches can occur when there is overlap between an anonymized

network and a public network, causing what they call an *auxiliary network attack*. They used two large public networks, Twitter and Flickr, to demonstrate the results of an attack in this manner by anonymizing the Twitter graph and attempting to re-identify nodes based on the similarities in graph structure between the Twitter and Flickr graphs. Narayana and Shmatikov argue that the identification of "important" nodes is more damaging than nodes that are not important and therefore factor node importance into their success measure, using the degree of the node as a proxy for importance. Using this weighted measure, they were able to get a success rate of 30 %, indicating that the attack is a realized possibility.

3 Social Network Graph Anonymization

Due to the array of privacy breaches discussed in the previous section, a number of new techniques that build upon naïve anonymization have been introduced to better anonymize social network graphs. Many of these techniques are based on the concept of k-anonymity, which we discuss below before presenting the techniques themselves.

3.1 k-Anonymity

Samarati and Sweeney [26] proposed an anonymization technique that generalizes quasi-identifiers so that at least some minimum number of people, k, have the same quasi-identifier values. A quasi-identifier is a piece of information about a person that does not uniquely identify him/her, but does identify a characteristic about him/her, such as a birthday that can be used with other information to identify an individual. The value k can be arbitrarily set depending on the data set, with higher values of k indicating that the data set is more resistant to privacy breaches. Samarati and Sweeney formally defined k-anonymity as follows:

> Let $T(A_1, \ldots, A_n)$ be a table and $Q|T$ be the quasi-identifiers associated with it. T is said to satisfy k-anonymity iff for each quasi-identifier $QI \in Q|T$ each sequence of values in $T[QI]$ appears at least with k occurrences in $T[QI]$.

For example, in Table 1, the Hair Color variable would be considered k-anonymous with $k = 2$ because there are at least 2 people with each value of Hair Color. However, the Eye Color variable would not be k-anonymous with $k = 2$ because not all values, i.e. brown, are associated with at least 2 people.

There are a number of options for making a data set k-anonymous. In the example above, one option would be to not publish either the Eye Color column or the user Ted. Alternatively, we could change the non-k-anonymous attribute to match another attribute that is k-anonymous, e.g. change Ted's eye color to Green, or change a different attribute to make all attributes k-anonymous, e.g. change Bill's eye color to brown. The example above applies k-anonymity to relational data, but we can

Table 1 Hair and Eye Color

Name	Hair Color	Eye Color
James	Brown	Green
Amy	Brown	Green
Bill	Blond	Green
Ted	Blond	Brown

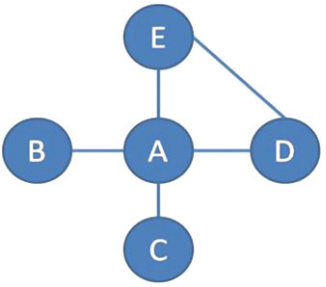


Fig. 4 A simple network graph

also apply it to network data. In Fig. 4, there are five nodes connected to each other in a variety of ways. We can apply k-anonymity by looking at the degree of each node. Nodes B and C are each connected to only one other node, nodes D and E are each connected to two other nodes, and node A is connected to all four other nodes. Again assuming that $k = 2$, nodes B, C, D, and E are k-anonymous, but node A is not since it is the only node that is connected to four nodes. k-anonymity has been extended to consider the importance of non-identifying information via l-diversity [20], t-closeness [18], and k-symmetry [30]. However, we will focus on k-anonymity as the primary anonymity approach since it is the primary technique used for social network anonymization.

3.2 Anonymization Techniques

Due to the privacy breaches discussed above, a number of new techniques for anonymizing social network graph data emerged over the past few years. In one of the earliest attempts at anonymization, Liu and Terzi [19] focused on anonymizing the degree of the nodes in the network. Liu and Terzi apply k-anonymity by ensuring that the degree of all nodes is k-anonymous, manipulating the graph to guarantee that there are at least k nodes for each degree existing within the full set of degrees for the network. They do this by gathering the degrees of all nodes in the graph, identifying which degrees do not have k nodes, and then changing the degree of those nodes to match the closest degree that is k-anonymous, or creating a new degree set that is k-anonymous from nodes that are not k-anonymous. After creating the new degree set, if the sum of all of the degrees is even, they construct a graph

based on it. However, if the sum of all degrees is odd, then a graph cannot be constructed, so they return to the original graph, add random noise, and re-anonymize until the sum of all degrees is even and a graph can be constructed. Rather than focusing on node degree, Zhou and Pei [33] directly address the neighborhood attacks presented by Backstrom, Dwork, and Kleinberg [4] by manipulating graphs to have k-anonymous subgraphs based on a measure of the local neighborhood graph for all nodes. Their method relies on adding edges to the graph to make nodes that have distinct neighborhoods similar to other nodes to improve the anonymity of the graph. Similarly, Zou, Chen, and Ozsu [34] attempt to alter subgraphs in a way such that all subgraphs are k-anonymous through a process entitled k-automorphism. k-automorphism focuses on anonymizing subgraphs rather than node neighborhoods, which allows this method to be dynamic as a graph changes. More recently, Cheng, Fu, and Liu [9] developed the k-isomorphism method for anonymization of social network data. k-isomorphism creates k isomorphic subgraphs to preserve privacy at the subgraph level.

Hay, Miklau, Jensen, and Towsley [16] take a different approach to anonymization they call *graph generalization*. Rather than attempt to keep the graph as close to its original form as possible, as the methods above do, Hay et al. partition portions of the graph and measure the topological properties of each partition. Additionally, Hay et al. record the connections between these partitions. The result is a graph that holds the properties of the original graph, but does not have the same granularity. More so than with other anonymization algorithms, the generalization method, while increasing anonymity, decreases the utility of the anonymized graph in many cases. Capman and Truta [8] offer a similar approach they call *masked social networks*, although in additional to generalizing the graph itself, they also generalize the profile information of users in an attempt to maintain maximum utility of the anonymized social network graph. He, Vaidya, Shafiq, Adam and Atluri [17] utilized a similar anonymization method that partitions the network in a manner that preserves as much of the structure of the original social network as possible. They do this by anonymizing the local structures of individual nodes such that all generalizations reflect actual structures of the original graph. Bhagat, Cormode, Krishnamurthy, and Srivastava [7] built upon the graph generalization method but added additional constraints on how the partitions occur in an effort to decrease the susceptibility to attack. They also introduce the concept of *label lists* as a potential anonymity mechanism. They use label lists to obscure the identity of a particular node by assigning it a number of labels, one of which is the real label. In this way, an attacker could not be certain which node he/she is examining; however, this diminishes the utility of the perturbed graph at a rate that correlates with the number of fake labels applied to each node. Singh and Schramm [27] take the generalization concept further and create a generalized trie structure that contains information about network subgraphs and neighborhoods. This information can be used to answer questions about network centrality characteristics without revealing sensitive information.

While most studies have focused on the attributes of the nodes and breaches of node identity, an additional avenue of attack is to infer the existence of an edge between nodes. This represents a breach in privacy when a node is able to determine

whether or not two nodes connected to it are connected. Zheleva and Getoor [31] present a number of different strategies for preventing edge inference that increasingly provide greater anonymity at the cost of decreased usability of the data. These strategies include removing all sensitive edges, removing a percentage of all edges, clustering nodes to hide true edges, clustering nodes in a constrained manner, and removing all edges completely. More recently, Das, Egecioglu, and Abbadi [10] presented an additional method for anonymizing social network graphs which focuses on the edges rather than the nodes in the graph. Their method utilizes a linear programming model and weighted edges in an effort to preserve as much utility in the anonymized graph as possible.

In addition to the graph modification and generalization methods mentioned above, differential privacy [11] has become an important foundation for social network anonymization methods. Differential privacy emphasizes that the structure of allowable queries on a statistical database must be designed such that a malicious attacker with the ability to query the database, but without direct access to the full database, cannot determine the unique characteristics of a specific individual. Since social network linkages can be considered a type of statistical information, differential privacy helps to prevent the de-anonymization of social network users while allowing for useful information via queries to the database [12]. While surveys of differential privacy literature are available [13, 14], we highlight three articles that have specific implications for social networks. First, Mironov, Pandey, Reingold, and Vadhan [22] show that the computational power of an adversary should be considered when designing differential privacy systems since this allows for a bounding mechanism that results in systems more appropriate to real-world attacker resources. Second, McSherry and Mironov [21] apply differential privacy methods to network recommendation systems, specifically the Netflix Prize. Finally, Hay, Liu, Miklau, Pei, and Terzi [15] provide a tutorial for applying differential privacy techniques to real-world databases in a manner that can allow the owners to have provable security for the results from queries to their database.

4 Measuring Graph Anonymity

All of the studies above have focused on social network graph anonymization techniques and do not have a comprehensive method for measuring how much anonymity is inherent in a network or how susceptible to privacy breaches an anonymized network is. Due to a need to measure networks, rather than anonymizing them without being able to quantify the improvement of anonymity levels, Singh and Zhan [28] developed a measure entitled *topological anonymity*. Topological anonymity measures the susceptibility of a given network topology to privacy breaches. Singh and Zhan focus on two specific types of privacy breaches, the node identity breach and edge inference breach, discussed in Sect. 2. The topological anonymity measure utilizes the network measures of degree and clustering coefficient to assess a network graph and determine the level of anonymity inherent in

the topological structure of the graph. To do this, they define D_a to be the set of nodes with degree a. This is first used to determine if there are enough nodes with the same degree so that the nodes are indistinguishable from at least some number ε other nodes. Next, to ensure that all the nodes with the same degree set do not have the same neighborhood structure, they introduce a Boolean measure called *CC_dif* and calculate it as follows:

$$CC\_dif_a = \begin{cases} 0 & \text{if } \text{var}(CC(D_a)) = 0 \\ 1 & \text{if } \text{var}(CC(D_a)) > 0 \end{cases} \tag{8}$$

where $\text{var}(CC(D_a))$ is the variance of the clustering coefficients for nodes of degree a in degree set D_a. Combining these two measures yields the following equation representing the topological anonymity, *ta*, of a connected network:

$$ta = \frac{\sum_{i=1}^{\max(\deg(G))} \left[(|D_i| * CC\_dif_i) - \begin{cases} 0 & \text{if } |D_i| \geq \varepsilon \\ |D_i| & \text{if } |D_i| < \varepsilon \end{cases} \right]}{n} \tag{9}$$

where ε represents the required number of nodes in a degree set to ensure a given level of anonymity in a graph. This number is chosen in advance, and a higher value of ε represents a higher level of required anonymity, similar to k-anonymity defined above. The resultant *ta* score is a simple number between -1 and 1 that summarizes the anonymity inherent in the graph, with -1 indicating the graph is highly susceptible to node identity and edge inference privacy breaches due to a large amount of distinctness of nodes due to degree and clustering coefficient, and 1 indicating the nodes and edges in the graph are well anonymized and not distinct.

After presenting the *ta* measure, Singh and Zhan proceed to provide a number of examples to demonstrate the usage of the measure. They utilize several small, hand-crafted datasets to show the various ranges of the measure, and then run the measure against one real-world dataset and two computer generated graphs. The real-world dataset is a crawl of political blog sites [3] containing 1224 nodes and an average degree of 27. Singh and Zhan randomly generated the two computer generated graphs, but designed one to follow an Erdös-Rényi distribution where the degree of the nodes follows a binomial distribution, and one to follow a scale-free distribution. Finally, Singh and Zhan compare the *ta* score of each of the graphs with various values of required anonymity, ε. This comparison shows that, as expected, the *ta* score is influenced heavily by the node degree distribution, and that the random binomial graph has a higher *ta* score, indicating it is more anonymous, than the political blog network graph and the scale-free graph. Since social networks have been shown to follow a power-law degree distribution [5], the *ta* score tends to be between approximately -0.2 and 0.8 when the standard value of $\varepsilon = 3$ is used, although this varies as ε increases [24].

Not only can the topological anonymity measure be used for measuring the anonymity inherent in naïvely anonymized social network graphs, but it can also be used to compare the affects of applying the anonymization techniques above. The topological anonymity measure can be applied to a social network graph before and after an anonymization algorithm is applied. The positive change in the *ta* score can

be used as a measure of the effectiveness of the anonymization algorithm and can then be compared to other anonymization algorithms. Nagle, Singh, and Gkoulalas-Divanis [24] show that topological anonymity is indeed a useful method for comparing different types of noise adding anonymization schemes. This process helps those looking to release social network graph data understand which anonymization algorithm will best anonymize their particular graph.

5 Conclusion

Social network graphs can provide a fascinating data set for researchers, but the release of such data must be done in such a manner that the privacy of the individuals in the network is maintained. While many methods of maintaining privacy have been presented above, they vary in their effectiveness of reflecting the properties of the original network graph, which is crucial for maintaining the usability of these network graphs for research purposes. Utilizing measures such as topological anonymity for assessing the levels of privacy and anonymity inherent in a network graph will allow us to compare the effectiveness of new anonymization techniques in preventing the various privacy breaches discussed above. Future research in this young field must build upon the existing research to offer anonymization methods that better maintain the usability of the network graph. Additionally, to stay ahead of attackers, researchers should continue to search for additional privacy breaches that bring to light new methods of obtaining sensitive information from social network graphs. Finally, further understanding of measuring the actual level of anonymity in social network graphs will allow for stronger and more efficient anonymization techniques that will allow for the secure release of larger real-world social network graphs for research purposes.

References

1. Acquisiti A, Gross R (2005, November) Information revelation and privacy in online social networks (the Facebook case). In: Proceedings of the ACM workshop on privacy in the electronic society (WPES)
2. Acquisiti A, Gross R (2007, October) Privacy risks for mining online social networks. In: NSF symposium on next generation of data mining and cyber-enabled discovery for innovation (NGDM '07)
3. Adamic L, Glance N (2005) The political blogosphere and the 2004 US election. In: Proceedings of the WWW-2005 workshop on the weblogging ecosystem
4. Backstrom L, Dwork C, Kleinberg J (2007) Wherefore Art Thou R3579X? Anonymized social networks, hidden patterns, and structural steganography. In: The 16th international world wide web (WWW) conference
5. Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
6. Barbaro M, Zeller T (2006, August 9). A face is exposed for AOL search no. 4417749. The New York Times

7. Bhagat S, Cormode G, Krishnamurthy B, Srivastava D (2009) Class-based graph anonymization for social network data. In: Proceedings of the 35th international conference on very large data bases (VLDB)
8. Campan A, Truta TM (2008) A clustering approach for data and structural anonymity in social networks. In: Proceedings of privacy, security and trust in KDD (PinKDD)
9. Cheng J, Fu A, Liu J (2010) K-isomorphism: privacy preserving network publication against structural attacks. In: Proceedings of the international conference on management of data
10. Das S, Egecioglu O, Abbadi A (2010) Anonymizing weighted social network graphs. In: Proceedings of the international conference on data engineering
11. Dwork C (2006) Differential privacy. Lecture notes in computer science, vol 4052, pp 1–12
12. Dwork C (2007) An ad omnia approach to defining and achieving private data analysis. In: Proceedings of the first ACM SIGKDD international workshop on privacy, security and trust in KDD (PinKDD)
13. Dwork C (2008) Differential privacy: a survey of results. In: Proceedings of the theory and applications of models of computation, 5th international conference (TAMC)
14. Dwork C (2009) The differential privacy frontier. In: Proceedings of the sixth theory of cryptography conference (TCC)
15. Hay M, Liu K, Miklau G, Pei J, Terzi E (2011) Privacy-aware data management in information networks. In: Proceedings of the 30th ACM SIGMOD symposium on principles of database systems
16. Hay M, Miklau G, Jensen D, Towsley D (2008) Resisting structural re-identification in anonymized social networks. In: Proceedings of the 34th international conference on very large data bases (VLDB)
17. He X, Vaidya J, Shafiq B, Adam N, Atluri V (2009) Preserving privacy in social networks: a structure-aware approach. In: Proceedings of the international conference on web on web intelligence and intelligent agent technology
18. Li N, Li T, Venkatasubramanian S (2007, April) t-closeness: privacy beyond k-anonymity and l-diversity. In: Proceedings of the international conference on data engineering (ICDE)
19. Liu K, Terzi E (2008) Towards identity anonymization on graphs. In: Proceedings of the special interest group on management of data (SIGMOD) conference
20. Machanavajjhala A, Gehrke J, Kifer D (2007) l-diversity: privacy beyond k-anonymity. ACM Trans Knowl Disc Data 1(1): Article No. 3
21. McSherry F, Mironov I (2009) Differentially private recommender systems: building privacy into the netflix prize contenders. In: Proceedings of the 15th ACM SIGKDD conference on knowledge discovery and data mining
22. Mironov I, Pandey O, Reingold O, Vadhan S (2009) Computational differential privacy. In: Proceedings of the 29th international cryptology conference, CRYPTO 2009
23. Nagle F, Singh L (2009, July) Can friends be trusted? Exploring privacy in online social networks. In: Proceedings of the 2009 international conference on advances in social network analysis and mining (ASONAM 2009)
24. Nagle F, Singh L, Gkoulalas-Divanis A (2012) EWNI: efficient anonymization of vulnerable individuals in social networks. In: Proceedings of the 16th Pacific-Asia conference on knowledge discovery and data mining (PAKDD)
25. Narayana A, Shmatikov V (2009) De-anonymizing social networks. In: Proceedings of the IEEE symposium on security and privacy
26. Samarati P, Sweeney L (1998) Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In: Proceedings of the IEEE symposium on research in security and privacy
27. Singh L, Schramm C (2010) Identifying similar neighborhood structures in private social networks. In: Proceedings of the IEEE International workshop on privacy aspects of data mining
28. Singh L, Zhan J (2007) Measuring topological anonymity in social networks. In: Proceedings of the 2007 IEEE international conference on granular computing
29. Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, Cambridge

30. Wu W, Xiao Y, Wang W, He Z, Wang Z (2010) k-symmetry model for identity anonymization in social networks. In: Proceedings of the 13th international conference on extending database technology, EDBT '10
31. Zheleva E, Getoor L (2007) Preserving the privacy of sensitive relationships in graph data. In: Proceedings of the first international workshop on privacy, security, and trust in KDD
32. Zheleva E, Getoor L (2009) To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In: Proceedings of the 18th international conference on the world wide web
33. Zhou B, Pei J (2008) Preserving privacy in social networks against neighborhood attacks. In: Proceedings of the 24th international conference on data engineering (ICDE)
34. Zou L, Chen L, Ozsu M (2009) Kautomorphism: a general framework for privacy preserving network publication. In: Proceedings of the 35th international conference on very large data bases (VLDB)

Partitioning Breaks Communities

Fergal Reid, Aaron McDaid, and Neil Hurley

Abstract Considering a clique as a conservative definition of community structure, we examine how *graph partitioning algorithms* interact with cliques. Many popular community-finding algorithms partition the entire graph into non-overlapping communities. We show that on a wide range of empirical networks, from different domains, significant numbers of cliques are split across the separate partitions produced by these algorithms. We then examine the largest connected component of the subgraph formed by retaining only edges in cliques, and apply partitioning strategies that explicitly minimise the number of cliques split. We further examine several modern overlapping community finding algorithms, in terms of the interaction between cliques and the communities they find, and in terms of the global overlap of the sets of communities they find. We conclude that, due to the connectedness of many networks, any community finding algorithm that produces partitions must fail to find at least some significant structures. Moreover, contrary to traditional intuition, in some empirical networks, strong ties and cliques frequently do cross community boundaries; much community structure is fundamentally overlapping and unpartitionable in nature.

Keywords Community finding · Partitioning · Clustering · Network analysis

F. Reid (✉)
Clique Research Cluster, Complex Adaptive Systems Laboratory, University College Dublin, Dublin, Ireland
e-mail: fergal.reid@gmail.com

A. McDaid · N. Hurley
UCD School of Computer Science & Informatics, University College Dublin, Dublin, Ireland

A. McDaid
e-mail: aaronmcdaid@gmail.com

N. Hurley
e-mail: neil.hurley@ucd.ie

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_5,

1 Introduction

Groups of interacting entities can be considered as a complex system. It is popular to examine such systems in terms of the networks their component entities form, to gain insight into properties of the system as a whole. For example, the speed with which a contagion can spread through a system is partly determined by the topology of its underlying network. The way subgroups of entities interconnect is also important to investigate whether useful higher level abstractions—above the level of individual entities—exist in the systems we study. To find such structures, an extensive variety of algorithms have been developed, which attempt to find groups of nodes in the network that are structurally significant in some way; these groups are referred to in the literature as 'communities'. See Fortunato [11] for an extensive review of these algorithms, which we will refer to as *Community Finding Algorithms*, or *CFAs*.

CFAs have been put to a range of applications, across several domains. As CFAs are applied ever more broadly, it is important that the structures they find, and the consequences of the design choices that define them are well understood. Particular CFAs should not be assumed to work across all complex networks, merely because they have evaluated well on some. In this research, we argue that certain algorithms, notably CFAs that produce *partitions* of the original network, return incomplete lists of the significant community structure, for at least some empirical networks. We perform an in-depth analysis of how several different CFAs interact with the cliques present in empirical networks, and discuss the consequences of this analysis for our intuition about community structure. We show that certain networks do not lend themselves well to partitioning, and caution against using partitioning algorithms as universal community finding tools.

1.1 Cliques as Lower Bound Communities

Each CFA finds structure that corresponds to a particular intuition of what a 'community' is; however there is little agreement on how exactly to define community. One common idea is that a community should have a high density of edges among its nodes, where *density* refers to the ratio of the number of actual edges between the nodes in the community to the maximum possible number of edges between these nodes. The bound of this definition is the graph theoretic structure known as a 'clique'—a fully connected subgraph, in which each node is connected to every other. Cliques, as discussed by Luce and Perry [22], have long been considered as community structure in human social networks. In the domain of social networks, this is particularly intuitive; if a user is friends with several others on Facebook, all of whom are also friends, then this is a significant structure of common friends. In addition to this intuitive appeal, cliques are rare structures in the networks we study; due to the strict requirement for each node to connect to every other, clique structure is unlikely to arise by chance in a sparse network. Cliques are thus important structures. However, to define communities solely as cliques is very strict and

conservative, for if even one connection in the group is missing—perhaps due to an incompletely observed network—then the found community will shrink. Many CFAs thus try to find communities comprised of groups of nodes which are highly connected, but less connected than perfect cliques. However, we posit that a clique is a good conservative *lower bound estimate* of community structure, in so far as an observed clique more than likely is wholly contained inside *some* real-world community. A maximally interconnected group of nodes, in a sparse network, always represents an interesting structure.

1.2 Partitioning Community Finding Algorithms

Many leading CFAs assign communities by partitioning the network, that is, grouping the nodes into *disjoint* subsets, assigning each node to exactly one subset. This partitioning approach to community finding has become popular, perhaps due to the appeal of treating a complex network as a graph, and the body of literature on graph partitioning problems. Early applications of graph partitioning, such as the applications of the Kernighan-Lin algorithm [16] discuss problems that explicitly require partitions, such as electronic component layout. However, in this work we are concerned about the completeness of the lists of community structure found by algorithms when used in other domains, such as social networks, and in complex networks generally. Regarding cliques as underestimates of community structure, we believe that regardless of what specific structures a given CFA finds, to be thorough, it should find, for each clique, *at least* one structure which is a superset of that clique. A CFA—considered as a tool that reveals structure in a complex network—that returns no community in which a group of fully connected nodes are assigned together, is neglecting to provide a complete list of the structures in the network. This is especially true if the clique is large in size.

1.3 Related Work

We show that in many complex networks, partitioning CFAs split cliques occurring within the network; and hence fail to find complete lists of the network structure. We examine why this occurs, investigating the intuition underlying many partitioning CFAs, and their relationship with cliques. We show, using cliques as a tool, that some traditional intuition describing communities as well connected sets of nodes, separated by narrow bridges, is not always correct. Instead, many of the graphs we study exhibit a structure that can be better explained as the 'pervasive overlap' discussed in [1, 9] than as independent, weakly-connected modules. We analyse cliques, rather than any other community structure proposed in the 'overlapping community finding' literature, because we require a definition of structure that is a fundamental, conservative, and convincing underestimate of community; for every community, we want to find a conservative subset of that community. We use

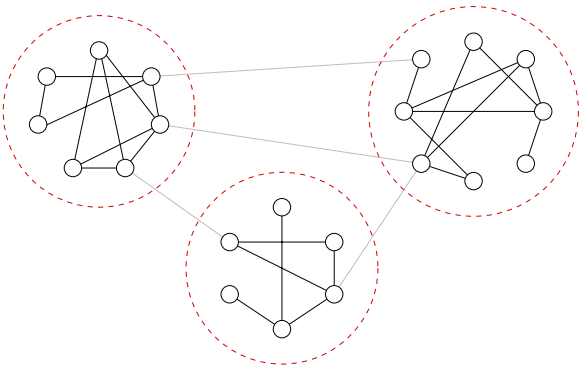


Fig. 1 Motivating image of network community structure from Newman [24]

cliques, rather than structures such as the percolated k-cliques of Palla et al. [25], because with percolated k-cliques, we find no universal k consistent across networks with which to evaluate partitioning; this would make it difficult to be conservative in our analysis. Rather than choosing a new definition of community and discussing whether it is sufficiently conservative, we instead use the fundamental definition of the clique and examine its implications in detail. We analyse some of the same data as Leskovec et al. [21]. However, while that influential work sought to investigate the quality of the best community structure, at each scale, by evaluating it in terms of *conductance*, which penalises communities in proportion to their external edges, we instead investigate network structure from a different angle, by using the sociologically grounded idea of the clique to conservatively estimate community *cores*. We characterise to what level each and every clique is preserved after the network is partitioned, thus considering structures globally across the network.

An illustration of the intuition behind many CFAs can be seen in Fig. 1, from the influential paper by Newman [24], which shows separate and well-defined modules, connected by only narrow bridges. This same intuition, conceptualising communities as connected by narrow bridges, can be traced back to the seminal work of Granovetter [13]: "*If the motivation to spread the rumor is dampened a bit on each wave of retelling, then the rumor moving through strong ties is much more likely to be limited to a few cliques than that going via weak ones; bridges will not be crossed.*"

Here, Granovetter is using 'clique' in the sociological sense, closer to the modern idea of community, and the idea is that bridges—narrow connecting links—need to be crossed to carry information between such cliques. This idea is further summed up in the modern review of Fortunato [11] as: "*If it were possible for a clique to move on a graph, in some way, it would probably get trapped inside its original community, as it could not cross the bottleneck formed by the inter-community edges.*" However, this work, in keeping with research [28, 30] on a limited number of other networks, finds evidence that structurally weak ties need not be crossed to traverse the network, contrary to the intuition just described. In fact, we show that while the traditional intuition may be appropriate in some cases, the structure of many

empirical networks does indeed lead to cliques crossing the 'bottleneck' formed by inter-community edges.

2 Experiments

We conducted experiments to investigate the extent to which commonly used partitioning methods split the cliques in empirical network datasets. To keep the number of cliques we consider tractable, and in keeping with the original sociological definition of clique [22], we constrain our analysis to *maximal* cliques, which are cliques fully contained within no larger clique. For convenience, we refer to maximal cliques as simply 'cliques' in this work. In our analysis, we first generate the complete list of cliques present in each network using the fast Bron Kerbosch algorithm [3]. We then use the partitioning method under evaluation to assign each node to a community, and characterise how the cliques interact with the partitions found. We examine each maximal clique in turn, checking whether it is fully contained within a partition, or to what extent it has been split across partitions. We quantify and present this metric for each network, initially using two distinct partitioning methods; one popular and efficient modularity optimisation method [2] and one normalised min-cut optimising method [6].

2.1 Network Datasets Examined

To analyse data from a wide variety of networks, we gathered data from several different sources. We used several network datasets from the SNAP project[1] [21]. We examined networks formed by patterns of communication: The Enron and EU E-mail networks, and mobile telecoms data provided by an industrial partner,[2] comprised of the voice call and SMS interactions on a mobile telecoms operator. We examined relation networks formed in online social networks, consisting of several Facebook university network datasets [29], samples taken from the full Twitter follower network [17], and the Slashdot online network. For both Twitter and the Mobile telecoms data, where we had access to very large networks, beyond reasonable computational means to analyse, we generated 3 random snowball samples of each network to produce tractable datasets. For the Facebook datasets, we chose to run our experiments on the smaller networks, due to the computational cost of calculating all maximal cliques. We also considered the SNAP academic publication networks, the Web networks of Stanford and NotreDame, product recommendation networks from Epinions and Amazon, and Wikipedia voting network. Finally, we considered a Protein-Protein interaction (PPI) network [5], as an example of a biological network.

[1] http://snap.stanford.edu/data/

[2] Idiro Technologies.

2.2 Partition by Modularity Maximisation

Many of the most popular CFAs are based on the modularity maximisation approach of Newman [24]. The modularity function measures community quality as a count of internal edges, less the expected number in a random graph with the same node degrees. Modularity maximisation algorithms, such as the fast 'Louvain' method of Blondel et al. [2]—which we evaluate here—designed to have a low computational cost on sparse graphs, and scale to large mobile call networks—optimise for the number of partitions as well as the associated partitioning. While traditional intuition holds that even triangles, or 'strong ties', should not cross community boundaries, we are interested in more significant cliques—so we initially restrict our analysis to cliques of size at least 4. We also use a conservative definition of when a clique is 'split'—we say a clique is "split at level α" if no partition contains more than $(100 \times \alpha)$ % of its nodes. We quantify the proportion of cliques that are split by the partitioning of each network in two ways. First, we examine the proportion of cliques of size at least 4 that are split at level $\alpha = 0.9$. Table 1 shows the significant proportions of cliques split at this level. We would have expected, based on the traditional intuition, that such structures would be contained in the centre of the found communities—not spanning them, and not split by partitions that define found communities. Figure 2 provides an example of this effect, showing a single 4-clique that has been split across 4 separate partitions by the community finding algorithm.

As our metric is the proportion of maximal cliques that have been split, we might be concerned that many of the maximal cliques will be small, such as 4-cliques, and that if a 4-clique is split by partitions—while contrary to the intuition of structurally strong ties being unable to cross community boundaries—this might not be of particular concern. For a more conservative experiment, we consider only large cliques of size at least 8, split at level $\alpha = 0.8$. These parameters are arbitrary and we do not seek to justify them other than to reiterate that we are considering conservative structure, which would traditionally be expected only in the 'cores' of found communities, not on their boundaries—structure that a comprehensive CFA should return. Even with this conservative definition, the partitions break significant numbers of such structures, on many networks—see Table 1. For example, this shows that the Louvain CFA, run on the Caltech Facebook network, will split over one quarter of cliques of size 8 or more.

Our results show that the proportion of cliques split varies across the networks. There is also a large variation in the number of maximal cliques present. We might reason that this is due to some fundamental difference in the nature of the networks being considered, and question whether such analysis can be meaningfully applied across a range of networks. After all, the Amazon network is a network of frequently co-purchased products, and the web datasets are explicitly constructed lists of hyperlinks; still other networks involve human communication or collaboration. These networks are, however, frequently treated together as *complex networks*; we might *a priori* expect the same CFAs to perform well across them, and assume that a CFA proven in one domain will be automatically suitable and work well in other domains. However, this modularity method seems to do poorly on some types of network, at

Table 1 Proportion of maximal cliques split by the Louvain CFA, per network. We show the proportion of maximal cliques, of size 4 or greater, that have more than 10 % of their nodes assigned to different partitions (i.e. are split at level $\alpha = .9$). 'Prop large cliques split' is the proportion of maximal cliques, of size 8 or greater, that have more than 20 % of their nodes assigned to different partitions (i.e. are split at level $\alpha = .8$). 'Cliques' is the number of maximal cliques in the network, 'Partitions' is the number of partitions made by the Louvain method, and 'Largest Clique' is the size of the largest clique in the network

Network	Nodes	Partitions (as found by Louvain Method)	Maximal cliques	Largest clique	Prop. Cliques split	Prop. Large cliques split
Email-Enron	36,692	1,363	205,712	20	0.61	0.47
Email-EuAll	265,009	15,743	93,267	16	0.82	0.67
Mobile1	10,001	182	1,550	10	0.97	0.00
Mobile2	10,001	124	3,538	10	0.90	0.00
Mobile3	10,001	86	951	9	0.88	0.00
Facebook-Caltech	769	10	31,745	20	0.68	0.27
Facebook-Princeton	6,596	21	1,286,678	34	0.44	0.22
Facebook-Georgetown	9,414	26	1,440,853	33	0.41	0.22
Twitter1	2,001	8	23,570	12	0.99	0.66
Twitter2	2,001	4	554,489	27	0.15	0.01
Twitter3	2,001	7	130,399	22	0.06	0.00
Slashdot0811	77,360	771	441,941	26	0.13	0.01
Collab-AstroPhysics	18,771	331	27,997	57	0.60	0.32
Collab-CondMat	23,133	626	8,824	26	0.42	0.15
Collab-HighEnergy	9,875	483	2,636	32	0.23	0.00
Cite-HighEnergy	27,769	172	419,942	23	0.30	0.06
Amazon0302	262,111	173	117,054	7	0.01	0.00
Epinions	75,879	1607	1,596,598	23	0.38	0.11
Web-NotreDame	325,729	693	130,965	155	0.04	0.00
Web-Stanford	281,903	1013	774,555	61	0.04	0.01
Wiki-Vote	7,115	30	436,629	17	0.65	0.37
Protein-Collins	1,622	212	4,310	33	0.16	0.08

least where finding complete lists of community is desired. Similar results hold if we consider just the proportion of n-cliques split; as discussed in Sect. 3.2.

2.3 Relation of Modularity Found to Proportion Split

To investigate if the proportion of split cliques is in some way an artefact of low inherent modularity within the networks, we create a scatter-plot of the modularity

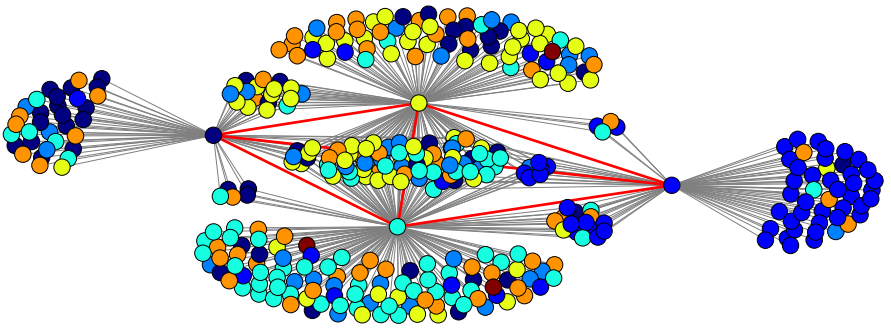


Fig. 2 Visualisation of one of the split 4-cliques from the Caltech Facebook dataset. Clique edges are shown in *red*; modularity partitions, as found by the Louvain method, are shown by colour; as can be seen, each node of this 4-clique has been assigned to a different community. This clique will thus not show up in the list of found communities. Note the many paths of length 2 between the clique's nodes. (Color figure online)

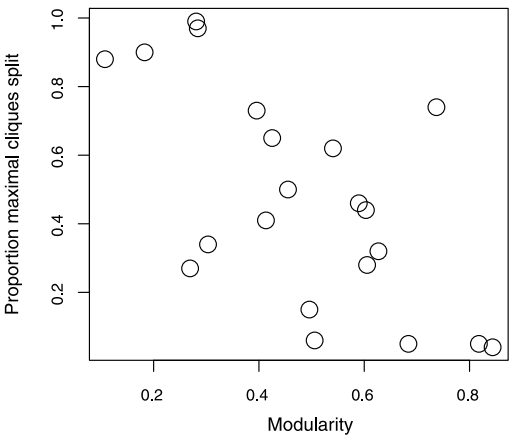


Fig. 3 Scatter plot of modularity of the partition vs the proportion of maximal cliques >10 % split (i.e. $\alpha = .9$), for each network

achieved, against the proportion of maximal cliques split. From Fig. 3 no obvious relationship appears. Several of the network partitions have high modularity and still display significant clique splitting; if there is a fundamental characteristic that renders particular networks unsuitable for modularity based partitioning, in terms of the proportion of cliques that will be split, then the modularity achieved does not capture it.

2.4 Partition by Normalised Edge Cut

Another method that has previously been used for the purpose of community finding, from a different family of algorithms, is the multilevel kernel k-means partitioning method implemented in *Graclus* [6], that minimises a normalised min-cut

Table 2 Proportion of cliques of size at least 4, split more than 10 % (i.e. $\alpha = .9$), by *Graclus* [6], and *hMETIS* [15], per network. Values shown for 4, 16 and 64 Partitions, with *ufactor* 50, and 16 Partitions with *ufactor*500. Also shown, proportion of the large connected component preserved, for subgraphs of edges in at least 4-Cliques '4-Clique', and edges in at least 5-Cliques '5-Clique'

Network	Graclus			hMETIS			16	4-Clique	5-Clique
	4	16	64	4	16	64	*uf*500		
Email-Enron	0.38	0.74	0.92	0.10	0.54	0.67	0.38	0.55	0.39
Email-EuAll	0.53	0.86	0.98	0.20	0.58	0.76	0.42	0.04	0.02
Mobile1	0.75	0.88	0.99	0.47	0.81	0.93	0.80	0.17	0.07
Mobile2	0.66	0.93	0.97	0.47	0.77	0.92	0.64	0.20	0.09
Mobile3	0.83	0.93	0.96	0.48	0.82	0.95	0.77	0.06	0.01
Facebook-Caltech	0.62	0.86	1.00	0.56	0.89	0.99	0.57	0.89	0.84
Facebook-Princeton	0.33	0.69	0.89	0.32	0.58	0.89	0.36	0.92	0.89
Facebook-Georgetown	0.30	0.58	0.80	0.32	0.50	0.74	0.40	0.93	0.90
Twitter1	0.88	0.99	1.00	0.82	0.97	1.00	0.83	0.78	0.57
Twitter2	0.22	0.99	1.00	0.05	0.88	1.00	0.56	0.70	0.57
Twitter3	0.74	0.98	1.00	0.04	0.65	0.99	0.05	0.74	0.33
Slashdot0811	0.28	0.49	0.94	0.08	0.13	0.37	0.09	0.10	0.04
Collab-AstroPhysics	0.43	0.53	0.77	0.27	0.49	0.65	0.34	0.83	0.71
Collab-CondMat	0.28	0.40	0.50	0.17	0.30	0.39	0.30	0.71	0.52
Collab-HighEnergy	0.16	0.28	0.43	0.10	0.19	0.29	0.19	0.42	0.13
Cite-HighEnergy	0.13	0.35	0.55	0.15	0.31	0.47	0.30	0.75	0.62
Amazon0302	0.01	0.02	0.04	0.00	0.00	0.00	0.00	0.11	0.00
Epinions	0.46	0.88	0.81	0.24	0.51	0.63	0.30	0.18	0.12
Web-NotreDame	0.01	0.03	0.11	0.00	0.05	0.18	0.04	0.07	0.03
Web-Stanford	0.03	0.04	0.39	0.00	0.09	0.46	0.02	0.49	0.40
Wiki-Vote	0.48	0.96	1.00	0.51	0.88	0.99	0.51	0.43	0.35
Protein-Collins	0.00	0.16	0.93	0.00	0.79	0.95	0.01	0.59	0.36

objective. Like the modularity maximisation method of Blondel et al. this implementation is designed to scale to large networks, performing well on sparse data by avoiding expensive eigenvector computation.

We examined this method on the same data as the modularity maximisation method. Unlike the modularity method, which discovers the number of partitions into which to break a graph, *Graclus* requires this to be specified. All other things equal, we would expect a smaller number of partitions would result in a smaller proportion of the maximal cliques being broken, and this effect is visible. However, even when asked to produce a relatively small number of partitions—relative to the network sizes—min-cut partitioning results in large proportions of the cliques greater than size 4 being split on many datasets, as shown in Table 2.

3 Fundamental Partitionability of Networks

Some datasets have a higher proportion of cliques split by partitions than others. This is largely uncorrelated with the mere number of cliques in the dataset, or the number of cliques per node, or per edge, or a number of other simple graph measures, such as clustering coefficient. After investigating several popular CFAs, we now consider whether any partition exists which would not split cliques. Perhaps there were potential partitions that would confine cliques to the cores of the communities found, but these methods were not finding them? To answer this, we consider, for each network, the subgraph induced by nodes that share cliques; i.e. the network formed by discarding all edges from the network that are not part of cliques. The connected components in this subgraph are the sets of nodes that cannot be placed into separate partitions without splitting any cliques. We calculate the size of the largest connected component of each network, and present this as the proportion of nodes in the network, in Table 2.

We show results for the subgraph induced by nodes that share cliques of size 4 or greater, and of size 5 or greater, under the headings '4-Clique' and '5-Clique'. On some networks, such as Facebook, Twitter, or collaboration networks, any partitioning scheme that is constrained to not split cliques of size five or greater has to leave the majority of nodes in a single partition.

This is an important structural property of these datasets, and an important result for certain diffusion models of complex contagion [4] which can only spread over structurally strong ties, as it shows these graphs are connected when using solely strong ties—it is possible to walk the graph communities without using weak ties. Further, on some of the larger datasets such as the Slashdot dataset, with 77,360 nodes, we find that over 30 per cent of those nodes (23,980 nodes) are in a connected component of the subgraph containing only edges that are in *triangles*; further evidence against the strict idea that strong ties do not cross community boundaries, and that communities are well separated.

3.1 Partitions that Directly Minimise Clique Splits

Having established the limits of partitions that break no *single* clique, we consider partitioning to directly optimise the number of cliques preserved, while producing balanced partitions. Partitioning a network while splitting as few cliques as possible is a hypergraph partitioning problem, where nodes in a clique together are connected by a hyperedge. This simple observation enables us to use a balanced mincut hypergraph partitioning algorithm, such as implemented by *hMETIS* [15] to partition the graph, while directly minimising clique splitting. *hMETIS* requires an important parameter to determine partition balance. Too high a value results in trivial partitions, with the vast majority of nodes in a single partition; too low might force *hMETIS* to make more aggressive hyperedge cuts than is reasonable. We initially set this *ufactor* at 50 (meaning the largest partition may have 50 % larger weight than the average), to allow some unbalance. We examine cuts into 4, 16, and 64 partitions—generally

fewer partitions than the modularity maximisation approach finds on these graphs. We also present results for 16 partitions with *ufactor* 500, allowing very large variation in partition size.

The results are shown in Table 2. Partitions directly minimising clique split indeed result in reduced proportions of the cliques split, compared to the balanced mincut of *Graclus*. As the number of partitions, and balance between partitions, constrain *hMETIS* more than the modularity maximisation method, the results are not directly comparable. However, as this method is directly minimising clique cut, it should approach a lower bound attainable by any partitioning CFA, for the given number of partitions—and, with generous balance parameters, indeed does better than modularity maximisation.

Even so, partitioning the network using this method, on a range of datasets—notably the collaboration networks, the Wiki voting data, the telecoms data and especially the Facebook social networks—still results in substantial proportions of cliques being split, demonstrating the fundamental global unpartitionability of some networks.

3.2 Detailed Analysis of Sample Networks

We now present some detailed statistics from three arbitrarily chosen sample networks: the Princeton Facebook network, which we will look at in detail as a case study, and one of each of the mobile and twitter sample networks. This Princeton Facebook network with over 6,500 nodes is large enough to allow us meaningfully investigate medium and large scale community structure. Facebook network data is also relatively dense, in that it captures many long term social relationships for each user; this is in contrast to more fleeting, or partial, network information we might obtain by extracting a network from a short term snapshot of a communications network.

In Fig. 4 we show the number of cliques of each size in the network. We also show, for each clique size n observed in the network, the number of split cliques of that size. We plot this profile of cliques split, at each size, for each partitioning method investigated (as well as for non-partitioning Overlapping Community Finding Algorithms, which will be discussed in Sect. 4). We also show the proportion of cliques of size n split, for each value of n. We present results for three definitions of 'split'—where we consider cliques split if (b) any of their nodes have been partitioned from them, (c) greater than 10 % of their nodes have been partitioned from them, and (d) greater than 20 % of their nodes have been partitioned from them.

The Louvain method finds 21 partitions on this network; we use *Graclus* and *hMETIS* to produce the same number of partitions as the Louvain method. While the absolute number of cliques split tends to decrease as the metric becomes increasingly conservative, we note that in all cases, non-trivial numbers of cliques are split. As we would expect—as all partitioning algorithms try, in some sense, to avoid cutting edges—often the larger a clique is in size, the smaller the probability of the partitioning algorithms splitting it; however, cliques of all sizes are still split

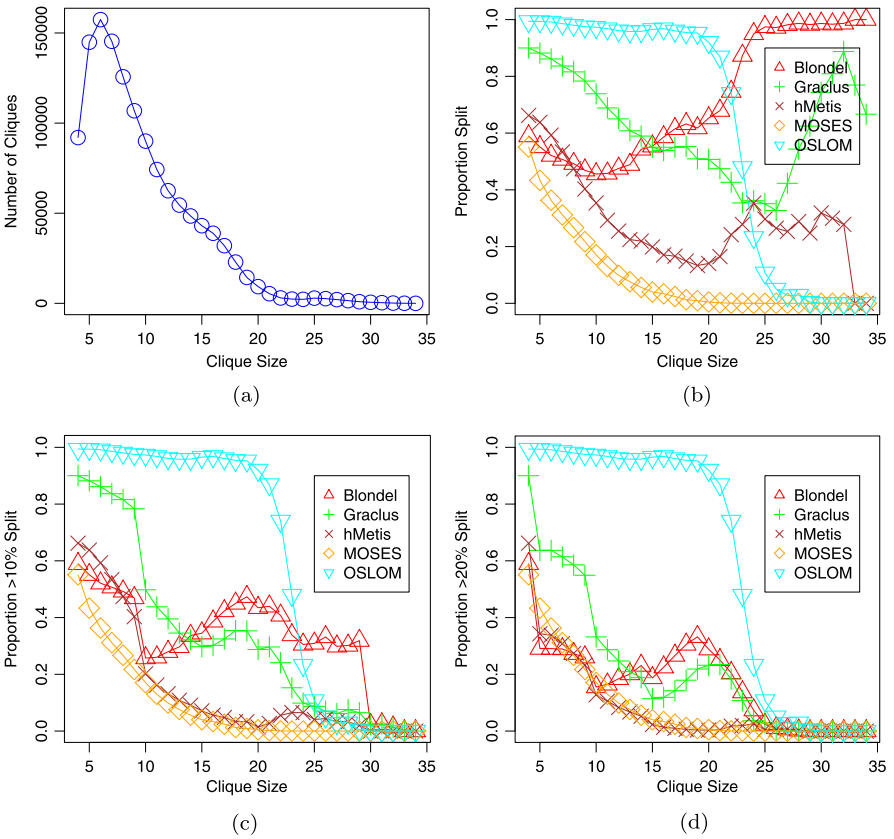


Fig. 4 Proportion cliques split at each size, for Princeton Facebook Network. (**b**) Considering a clique split if a single node is partitioned from it. (**c**) Considering a clique split if >10 % of its nodes are partitioned from it (i.e. $\alpha = .9$). (**d**) Considering a clique split if >20 % of its nodes are partitioned from it (i.e. $\alpha = .8$)

by these methods, on some networks; it is not the case that only the smallest cliques are split.

These figures emphasise the robustness of our findings—cliques of all sizes are split by partitioning—and illustrate an interesting way of characterising the effects of partitioning a network. Figures 5 and 6 show similar results for one of each of the Twitter and Mobile networks.

3.3 'Distinct' Cliques

A large clique, with some small portion of random edges deleted, will turn into many very similar smaller cliques. In quantifying the 'proportion' of cliques split, we might be concerned that mis-assignment of a small set of nodes, if they are

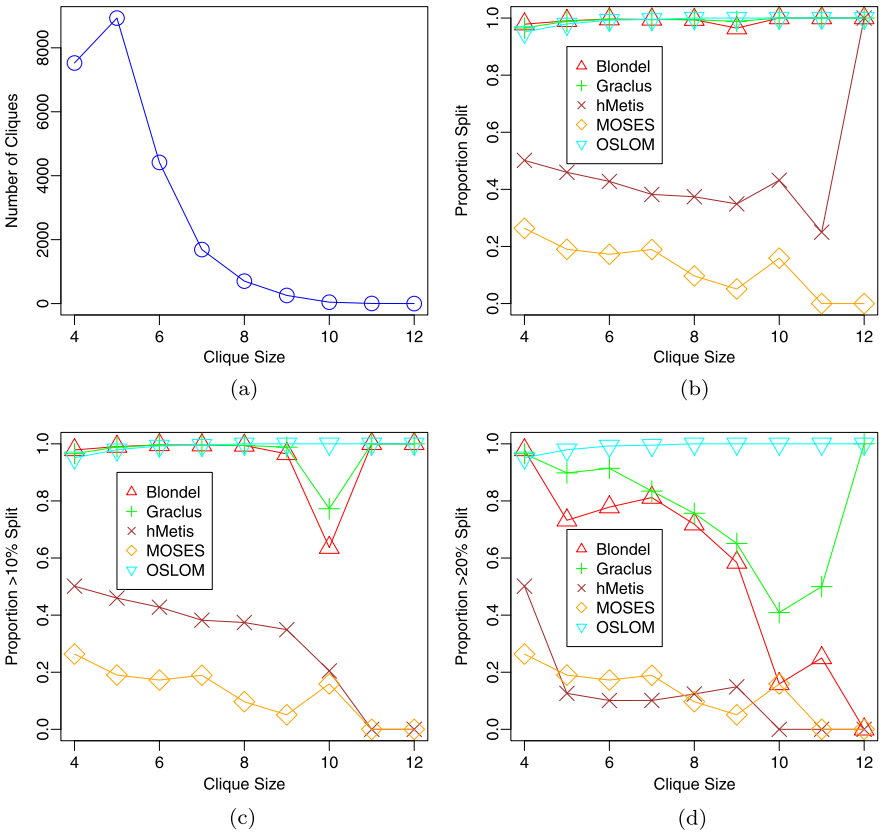


Fig. 5 Proportion cliques split at each size, for one of the Twitter Networks. (**b**) Considering a clique split if a single node is partitioned from it. (**c**) Considering a clique split if >10 % of its nodes are partitioned from it (i.e. $\alpha = .9$). (**d**) Considering a clique split if >20 % of its nodes are partitioned from it (i.e. $\alpha = .8$)

Table 3 Proportion of cliques that are 'distinct', beyond a given Jaccard similarity, that are over 10 % split (i.e. $\alpha = .9$) by *Graclus* [6], and *hMETIS* [15]. Values shown for *Graclus* and *hMETIS* for 16 partitions. *hMETIS ufactor* is 500

Network	Louvain	Graclus	hMETIS
Email-Enron	0.62	0.76	0.39
Mobile1	0.97	0.88	0.80
Facebook-Caltech	0.78	0.90	0.66
Twitter1	0.99	0.99	0.83
Collab-HighEnergy	0.23	0.28	0.19
Protein-Collins	0.34	0.33	0.02

contained within a large number of very similar overlapping cliques, might skew the proportions. As an additional check on the robustness of these results, we present a set of results in Table 3 which correct for this effect by running our analysis not on

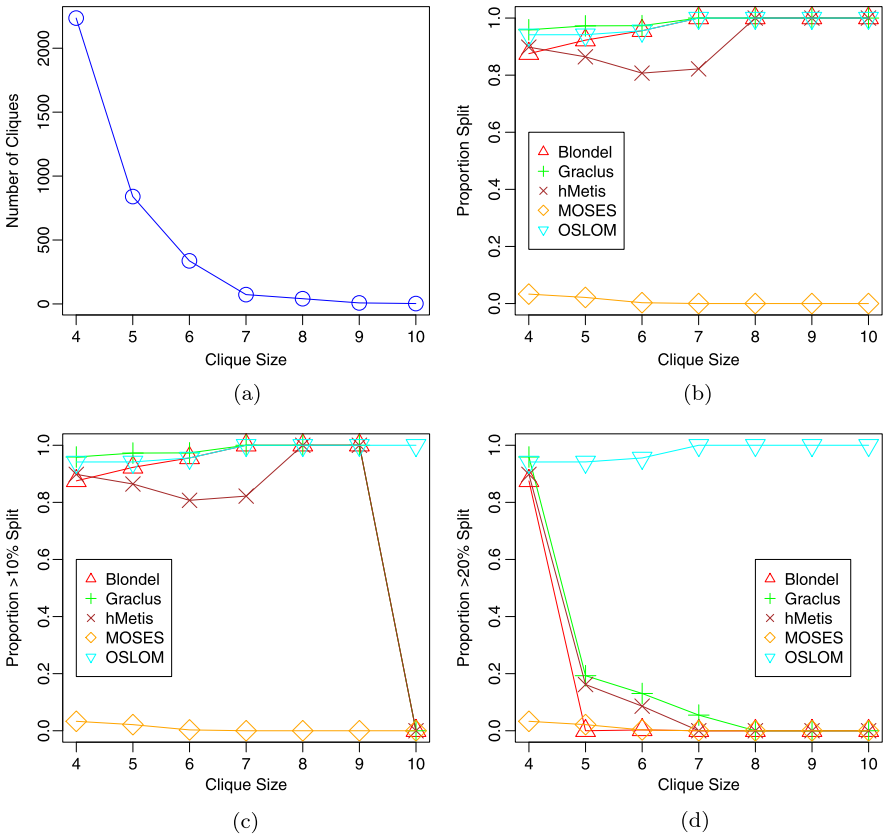


Fig. 6 Proportion cliques split at each size, for one of the Mobile Networks. (**b**) Considering a clique split if a single node is partitioned from it. (**c**) Considering a clique split if >10 % of its nodes are partitioned from it (i.e. $\alpha = .9$). (**d**) Considering a clique split if >20 % of its nodes are partitioned from it (i.e. $\alpha = .8$)

the full set of maximal cliques, but instead on a set of maximal cliques, after a pre-processing phase which removes any clique that has a high Jaccard similarity (>0.8) to any other larger clique. This analysis is computationally expensive to compute on the larger networks; however, on the networks we are able to perform it on, we find that our results still hold: substantial proportions of cliques are split, even if the only cliques we are looking at are cliques that are somewhat distinct from each other.

3.4 Random and Synthetic Models of Community

Broad categories of random community assignment model will produce networks where partitioning will fail to recover full communities. One source of

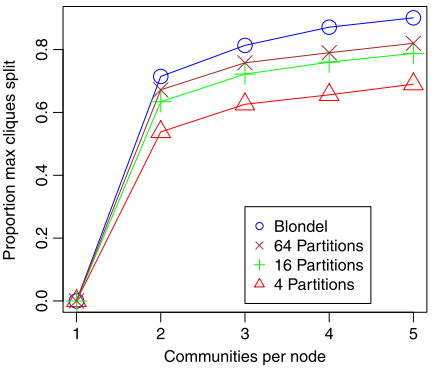


Fig. 7 Number of communities-per-node, as specified in benchmark parameters, vs proportion of maximal cliques >10 % split (i.e. $\alpha = .9$), by the Louvain and hypergraph partitioning methods, on LFR benchmark data. Each data point is the mean of 5 LFR instances; deviation is negligible

synthetic benchmark community data is the 'LFR' benchmark [18], in which 'communities'—defined as sets of nodes with a high probability of edges between them—are embedded into a generated network. We ran our experiments on LFR graphs to test our method on synthetic data. We generated realisations of a 10,000 node network, altering the number of communities each node was assigned to—from one to five, also increasing the corresponding number of edges, using the same parameters as with benchmarks detailed in previous work [20]. The results detailing the proportion of cliques split are shown in Fig. 7.

From these results, all methods partition the single-community-per-node networks without splitting cliques, but split significant numbers of cliques on networks with two or more communities-per-node. Even though the synthetic network model isn't directly embedding cliques—just increasing edge density within communities—partitioning fails to find all structure, by our defined metrics, on synthetic networks where nodes are overlapping. Further, large components exist in the graph of edges in cliques in these generated networks. Not only are the individual nodes and communities overlapping as designed by the model; it is a *global* property of the network as a whole that *no* non-trivial partition exists which does not split cliques.

4 Overlapping Community Finding Algorithms

A variety of algorithms exist which find overlapping communities within networks; Fortunato [11] mentions several of these, but this is an active area of research, with new algorithms frequently being developed [31]. Like with partitioning CFAs, many of these algorithms find subtly different structures, as authors work from slightly differing assumptions as to what constitutes a 'good' community. As such it is difficult to interpret what the output of a specific overlapping community finding algorithm tells us about the fundamental structure present in a network; and so we have avoided this in our analysis thus far. However, an advantage of overlapping CFAs is that they generally find structures that are much less common than maximal cliques are; typically a single overlapping community will contain many maximal cliques,

many of which may differ only by a small number of nodes. As we have seen, a great number of cliques exist in the networks we examine; and while we can investigate aspects of network structure by using these cliques, the fact that so many of them exist brings some disadvantages; specifically, the 'clique graph'—the graph of cliques that overlap each other—is typically too large to work with.

In previous sections, we have considered the partitionability of networks, concentrating our analysis solely on cliques as the cores of community structure. The notion of cliques as community cores can be explicitly encoded in a community finding algorithm, both to produce communities that are disjoint, if disjoint cliques are enforced [32] or overlapping, if this criterion is relaxed [14, 25, 27]. Indeed, this is the approach of a family of overlapping CFAs, which use cliques as 'seeds' for communities, including the 'Greedy Clique Expansion' (GCE) algorithm [20], to which the authors of this work have contributed. The GCE method starts with maximal cliques as seeds and grows these seeds into communities using a local community quality measure. Thus, it will trivially produce communities in which there exists, for each maximal clique, at least one community that fully contains it. However, many other overlapping CFAs have kept with an 'edge density' notion of community quality, and find communities without any explicit modelling of cliques. Given the large numbers of cliques present in empirical networks, approaches that do not explicitly model cliques can have computational advantages over those that do. It is thus interesting to apply our clique based analysis to such algorithms and ask if density-driven community finding algorithms preserve clique cores, when communities are allowed overlap.

In this section, we will make use of overlapping CFAs for two separate purposes. First, we analyse two overlapping CFAs with the same procedure as the partitioning CFAs, in order to ascertain the extent to which they split cliques in practice. Second, we examine the *community overlap graphs* created by these CFAs, and use the results of these graphs to examine the effects of partitioning on these networks, given the community structure as found by these particular overlapping CFAs.

4.1 Algorithms Examined

We concentrate our analysis on two recent overlapping CFAs: MOSES (*Model-based Overlapping Seed Expansion*) which we proposed in [23] and OSLOM (*Order Statistics Local Optimisation Method*) [19]. These statistically motivated algorithms do not explicitly use cliques in their operation, and are finding recent application in network analysis, for example the work of Grabowicz et al. [12]. While the complexity of these algorithms is dependent on the structure present in the input networks, like the previously discussed methods, both of these algorithms provide implementations which use heuristic techniques to enable them to scale to large networks, this makes them suitable for our analysis. We first examine the outputs of these algorithms. Figure 8 shows the community size distributions for each of the CFAs we analyse, on the case study networks we presented in detail earlier. We

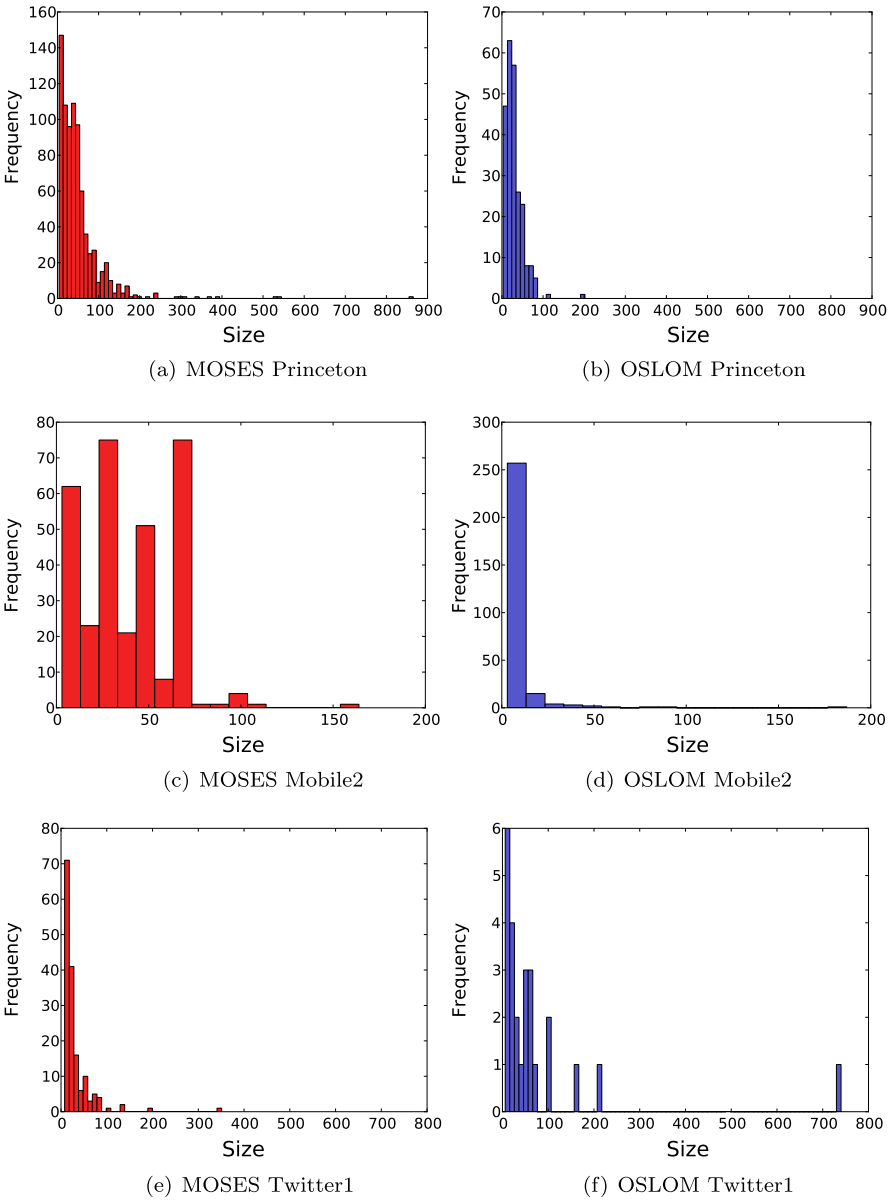


Fig. 8 Size distribution of overlapping communities found by MOSES and OSLOM. We do not show communities consisting of isolated nodes—OSLOM in particular finds a great many of these

present distributions, rather than average community sizes, as the distributions of community sizes found vary widely by network, and tend to be heavily skewed. We do not include 'singleton' communities, containing only single nodes; OSLOM in particular reports many of these.

4.2 Analysis in Terms of Split Cliques

We analysed the overlapping communities found by OSLOM and MOSES, in the same manner as the partitioning algorithms—for each clique, we check to see if there is any community in which it is fully contained; if there is not, we consider the clique to be split.

A thorough comparison of the exact structures found by these algorithms, each motivated by slightly differing models of community structure, is outside the scope of this work. To be thorough, we would have to either deal in subtle differences in the definition of 'community'—for which many definitions exist—or analyse the communities found by these methods against some 'ground-truth' data particular to a specific application domain. To restrict our analysis solely to network structure, we do not consider the issue of whether the communities found by MOSES and OSLOM are overall 'good' communities; instead we maintain our focus on split cliques. We present the results of this analysis in Table 4. We also show detailed results, for our case-study networks, on a per size-of-clique basis in Figs. 4, 5 and 6. These results show that MOSES produces a set of communities such that most larger cliques, in most networks, are contained in at least one community found by MOSES. This is an interesting result, considering that MOSES does not explicitly find communities in terms of cliques. The benchmarking of OSLOM yields a different result, however: for large numbers of cliques, OSLOM does not produce at least one community containing the clique.

It is difficult to explain this. Unlike MOSES, OSLOM outputs a hierarchy of levels of community, and we only consider the lowest level of that hierarchy; perhaps, in practice, the lowest levels are very 'fine grained' for OSLOM, below the level of an individual clique. Alternatively, in any community finding algorithm, there must always be a tradeoff between the sensitivity required to find all communities, and specificity to avoid finding 'false positive' communities. We can use cliques as underestimates of community structure, to measure sensitivity—in that every clique should be contained in a community—but not to measure specificity, for as discussed earlier, it may be too strict to require that every community contain a clique. Perhaps OSLOM is simply more specific in its output than MOSES; the two algorithms find structures of different quantity and size, as Fig. 8 shows. A detailed investigation of these issues would have to be undertaken in the context of a specific application domain, with ground truth data. But what these results do show is that at least some overlapping CFAs, which contain no explicit representation of cliques, find communities which split much fewer cliques than the partitioning algorithms do. The results also show that while the use of an overlapping CFA is necessary to avoid splitting cliques, as discussed in Sect. 3 and concretely illustrated by the results of MOSES, it is not a sufficient condition in practice as the OSLOM results show. Thus if a specific application domain requires high sensitivity, and a full list of community structure to be found, then not only must an overlapping CFA be used, but the CFA should also be evaluated for the specific application.

Table 4 Proportion of cliques that are not completely contained in at least one community—i.e. are 'split'—by the OSLOM and MOSES overlapping CFAs. Some networks present in previous benchmarks are not shown, due to the algorithms taking too long to run

Network	OSLOM >10 % split	OSLOM Size >8 >20 % split	MOSES >10 % split	MOSES Size >8 >20 % split
Email-Enron	0.96	0.98	0.16	0.01
Email-EuAll	0.93	0.00	0.06	0.00
Mobile1	0.99	0.00	0.00	0.00
Mobile2	0.94	0.00	0.03	0.00
Mobile3	0.99	0.00	0.03	0.00
Facebook-Caltech	0.76	0.37	0.41	0.04
Facebook-Princeton	0.93	0.76	0.21	0.01
Facebook-Georgetown	0.95	0.82	0.22	0.01
Twitter1	0.97	0.94	0.20	0.00
Twitter2	0.98	0.91	0.02	0.00
Twitter3	0.98	0.91	0.02	0.00
Collab-AstroPhysics	0.86	0.79	0.35	0.04
Collab-CondensedMatter	0.40	0.22	0.08	0.02
Collab-HighEnergy	0.33	0.00	0.08	0.00
Cite-HighEnergy	0.81	0.52	0.15	0.00
Amazon0302	0.09	0.00	0.01	0.00
Epinions	0.96	0.83	0.01	0.00
Wiki-Vote	0.79	0.67	0.04	0.00
Protein-Collins	0.13	0.06	0.06	0.01

4.3 Community Overlap Graphs

We have considered, in Sect. 3, the fundamental partitionability of networks, by examining the connected components which exist when we only consider subsets of the networks connected by cliques. We have also considered whether a hypergraph partitioning method, attempting to split as few cliques as possible, can partition the network, and have seen that—assuming cliques at the core of communities—communities overlap each other pervasively. We can develop our intuition about these results further, by considering the possibilities for the structure present in the '*Community Overlap Graph*' (COG); the graph formed by representing each community as a node, and connecting a pair of nodes (communities) by an edge, when the two communities overlap by more than some threshold number of nodes i.e. when they share more than some threshold number of nodes in common. When

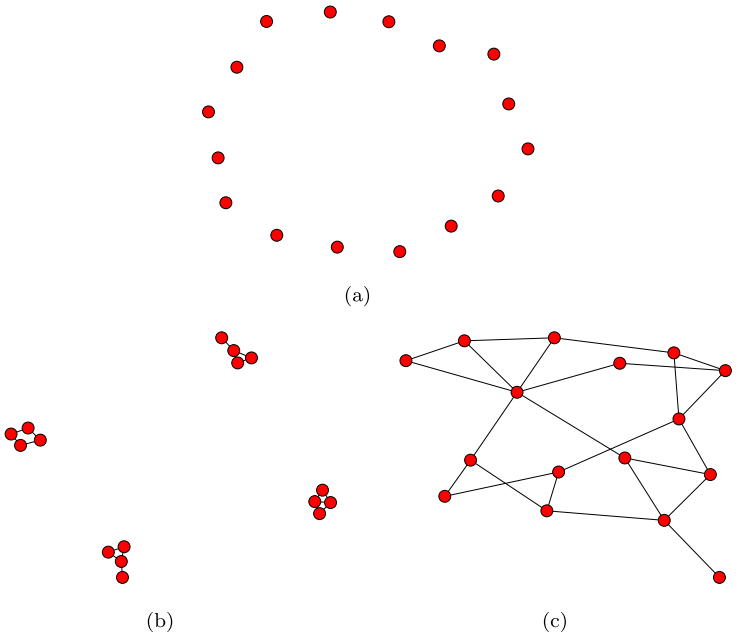


Fig. 9 Illustration of some possibilities for the Community Overlap Graph, for a network with 16 communities. Each node represents a community; edges connect pairs of overlapping communities. (**a**) Non-overlapping communities. (**b**) Overlapping communities, but clustered, with no path through overlap. (**c**) Overlapping communities with unpartitionable overlap

considering individual cliques as our communities, this idea is identical to that of the 'clique graph' discussed by Everett and Borgatti [10] and discussed further by Evans [8].

However, the large numbers of maximal cliques present in the networks we study make explicitly working with the clique graph difficult. The communities found by overlapping CFAs however, are typically smaller in number (as shown in Tables 5 and 6). Different possibilities for what structure we might see in the community overlap graph are shown in Fig. 9. We can see that Fig. 9(a) corresponds to a world view of non overlapping communities, in which the partitioning of networks into communities makes obvious sense. Figure 9(b) contains overlapping communities, but, perhaps surprisingly, it still makes some sense to partition the network, with partitions dividing clusters of overlapping communities together. We have shown, from our analysis of paths through cliques, and attempting to partition the network using hypergraph partitioning on the found cliques, that a world-view similar to Fig. 9(c) is most appropriate. We will now discuss these ideas in more detail, with reference to actual community overlap graphs, generated from the communities found by the two overlapping community finding algorithms, on empirical data.

4.4 Analysis of Community Overlap Graphs of Overlapping CFAs

In Fig. 10 we show visualisations of the Community Overlap Graph of the communities found by the MOSES and OSLOM algorithms, in the Facebook Princeton network. In order to show only the more significant overlaps, we draw an edge between two communities overlapping by at least 4 nodes. These visualisations show that in this particular network, most of the larger communities found by these two algorithms—and hence most of the nodes in the network—are part of a connected component of overlapping communities. As such, the empirical visualisation corresponds most closely to Fig. 9(c), and hence partitioning to find communities is not suitable in networks like this. Visualising the community overlap graphs of these networks shows clearly the extent to which communities overlap, and the structure that would be broken by partitioning these networks in order to find communities.

In addition to visualising these networks, we can attempt to quantify the degree to which a set of overlapping communities is partitionable, similar to how we examined the fundamental partitionability of networks in Sect. 3. We attempt to quantify this by examining how many of the communities in the community overlap graph are in the largest connected component of that graph, and what proportion of the nodes in the source network they contain. If a large proportion of communities and nodes are in a connected component, then this again would indicate quantitatively that the community structure is closer to Fig. 9(c) than (a) or (b).

Our results for the MOSES method are presented in Table 5, and for OSLOM in Table 6. As we can see, in line with our earlier results using cliques, the degree of overlap varies across networks with the social networks—particularly the Facebook networks—showing the least partitionable results.

In line with the results obtained by quantifying the proportion of cliques split, MOSES finds structure that is more highly overlapping than OSLOM. These results show an interesting method of quantifying the degree of overlap of community structure in a given network, and for a given overlapping CFA.

In Tables 5 and 6 we used an overlap of 4 nodes as a threshold for ‘significant’ overlap between two communities. It is interesting to examine how the threshold used to analyse the community overlap graph effects the connectivity of that graph. Figure 11 shows how the threshold effects the size of the largest connected component in the community overlap graph, both in terms of the number of communities in it, and the number of nodes of the underlying network that are in it, for both of the overlapping CFAs we examined. It can be seen from this figure that OSLOM has a sharp falloff in the size of the LCC as the threshold of overlap is increased. MOSES exhibits a much more gradual falloff in the size of the LCC—even if we require that communities have 10 nodes in common for them to be overlapping, the graph is still largely unpartitionable, without breaking several community overlaps. This difference between MOSES and OSLOM is perhaps not surprising, given MOSES’s tendency to find larger communities than OSLOM, and is consistent with the results in terms of the proportion of cliques split. This is further evidence that the level of overlap in the structures found by varying overlapping CFAs can vary widely.

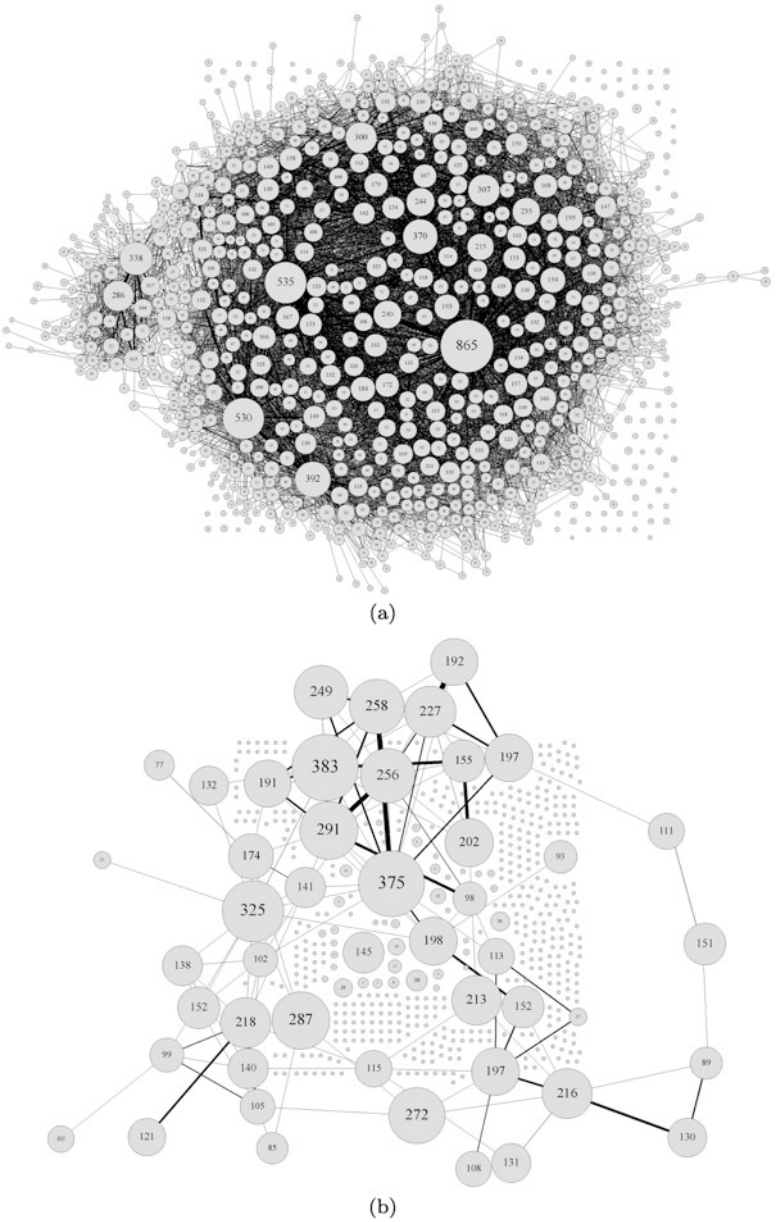


Fig. 10 Visualisation of COG of Facebook Princeton network. An edge is drawn whenever two communities overlap by at least 4 nodes. Edge width is proportional to overlap, and node area is proportional community size. Communities are labelled with the number of nodes they contain. Nodes may be present in multiple communities; two communities with a high degree of overlap contain fewer unique nodes than the sum of their labels. Shown here is the COG extracted from running (**a**) MOSES and (**b**) OSLOM on the Facebook Princeton network. Networks visualised with Graphviz [7] force directed layout

Table 5 Results for the size of the largest connected component (LCC) of the community overlap graph (COG), examining community structure found by MOSES

Network	Nodes	Number communities	Comms in LCC of COG	Nodes in LCC of COG	Proportion Nodes in LCC COG
Email-Enron	36,692	2,471	587	14,573	0.4
Email-EuAll	265,009	473	257	24,919	0.09
Mobile1	10,001	437	38	1,159	0.12
Mobile2	10,001	323	219	8,609	0.86
Mobile3	10,001	171	120	8,478	0.85
Facebook-Caltech	769	81	71	666	0.87
Facebook-Princeton	6,596	797	710	6,162	0.93
Facebook-Georgetown	9,414	893	800	8,740	0.93
Twitter1	2,001	161	132	1,686	0.84
Twitter2	2,001	129	99	1,680	0.84
Twitter3	2,001	188	101	1,080	0.54
Collab-AstroPhysics	18,771	2,816	677	9,953	0.53
Collab-CondMat	23,133	3,458	175	3,760	0.16
Collab-HighEnergy	9,875	1,663	15	274	0.03
Cite-HighEnergy	27,769	1,625	998	21,445	0.77
Amazon0302	262,111	23,665	47	1,767	0.01
Epinions	75,879	795	249	19,889	0.26
Wiki-Vote	7,115	65	63	3,805	0.53
Protein-Collins	1,622	150	16	221	0.14

5 Conclusion

We have investigated a wide range of empirical networks, characterising them according to the proportion of cliques in them that are split by various partitioning methods. Our results show that the early intuition on how communities are embedded in graphs does not hold across all networks and domains. On many complex networks cliques do not exist *solely* in community cores connected only by narrow bridges and weak ties—instead they frequently overlap across the community boundaries produced by partitioning algorithms.

If we accept cliques as conservative lower bounds for community structure, then, on many networks, partitioning CFAs are fundamentally limited in the completeness of the communities they can find, as shown by our results on the graph of edges in cliques, and from using hypergraph partitioning algorithms to partition cliques. This shows that communities are not easily separable from each other simply by remov-

Table 6 Results for the size of the largest connected component (LCC) of the community overlap graph (COG), examining community structure found by OSLOM

Network	Nodes	Number communities	Comms in LCC of COG	Nodes in LCC of COG	Proportion Nodes in LCC COG
Email-Enron	36,692	10,620	46	2,722	0.07
Email-EuAll	265,009	131,143	–	–	–
Mobile1	10,001	8,435	1	3	0
Mobile2	10,001	8,119	4	235	0.02
Mobile3	10,001	9,195	2	157	0.02
Facebook-Caltech	769	137	2	100	0.13
Facebook-Princeton	6,596	920	11	404	0.06
Facebook-Georgetown	9,414	1,189	5	178	0.02
Twitter1	2,001	467	21	1,529	0.76
Twitter2	2,001	113	7	1,463	0.73
Twitter3	2,001	289	9	1,040	0.52
Collab-AstroPhysics	18,771	4,106	9	202	0.01
Collab-CondMat	23,133	5,911	6	159	0.01
Collab-HighEnergy	9,875	3,808	7	145	0.01
Cite-HighEnergy	27,769	4,393	12	358	0.01
Amazon0302	262,111	37,374	19	424	0
Epinions	75,879	46,260	57	3,739	0.05
Wiki-Vote	7,115	2,744	21	4,085	0.57
Protein-Collins	1,622	529	2	50	0.03

ing structurally weak ties; instead, communities overlap across each other, with pairs of community frequently connected by strong ties, and other communities.

Our analysis of overlapping community finding algorithms has shown that some overlapping CFAs produce sets of communities in which each individual clique is fully contained. However, as we have shown, not all overlapping CFAs satisfy this property. We have presented community overlap graphs as another tool—in addition to cliques—with which to explore the output of overlapping CFAs. We have shown that on some networks, the communities as output by overlapping CFAs reveal a community structure that cannot be partitioned.

Thus, caution is warranted when using partitioning community finding algorithms where there is a sensitivity requirement that all significant community structure be found. In agreement with recent research on pervasive overlap, conceptualising networks as overlapping meshes of strong ties, with denser community regions, and using a CFA designed to find communities that overlap, will be more appropriate in many application domains.

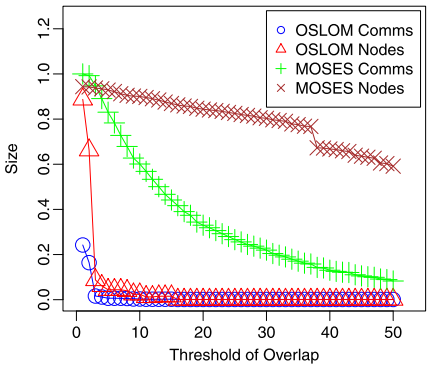


Fig. 11 As the threshold of overlap is changed, the size of the largest connected component in the community overlap graph changes. We investigate how the size of this component varies both in terms of the number of communities in it, and the total number of nodes connected by communities that overlap by at least that threshold. We display the size of the largest component, both in terms of the proportion of communities that are in it, and in terms of the proportion of nodes in the underlying graph which are in it, for both OSLOM and MOSES Overlapping CFAs

6 Further Work

Work on formal models of community generation that might explain whether a network is suitable for partitioning, and attempt to characterise the generative processes behind this global overlap would be interesting. That cliques frequently span communities also has implications for the type of diffusion processes that can occur on networks; data on the non-partitionable nature of communities may lead to an enhanced understanding of diffusion on complex networks. We hope that studying the nature of community overlap can lead to a better fundamental understanding of structure in empirical networks, and help development of future community finding algorithms.

Acknowledgements This work is supported by Science Foundation Ireland under grant 08/SRC/I1407: Clique: Graph and Network Analysis Cluster. An earlier version of this work appeared in *ASONAM '11* [26].

References

1. Ahn Y, Bagrow J, Lehmann S (2010) Link communities reveal multiscale complexity in networks. Nature 466(7307):761–764
2. Blondel V, Guillaume J, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008:P10008
3. Bron C, Kerbosch J (1973) Finding all cliques of an undirected graph. Commun ACM 16(9):575–577
4. Centola D, Macy M (2007) Complex contagions and the weakness of long ties. Am J Sociol 113(3):702–734

5. Collins S, Kemmeren P, Zhao X, Greenblatt J, Spencer F, Holstege F, Weissman J, Krogan N (2007) Toward a comprehensive atlas of the physical interactome of Saccharomyces cerevisiae. Mol Cell Proteomics 6(3):439
6. Dhillon I, Guan Y, Kulis B (2007) Weighted graph cuts without eigenvectors: a multilevel approach. IEEE Trans Pattern Anal Mach Intell 1944–1957
7. Ellson J, Gansner E, Koutsofios E, North S, Woodhull G (2004) Graphviz and dynagraph static and dynamic graph drawing tools. In: Graph drawing software
8. Evans T (2010) Clique graphs and overlapping communities. J Stat Mech Theory Exp 2010:P12037
9. Evans T, Lambiotte R (2009) Line graphs, link partitions, and overlapping communities. Phys Rev E 80(1):016105
10. Everett M, Borgatti S (1998) Analyzing clique overlap. Connections 21(1):49–61
11. Fortunato S (2010) Community detection in graphs. Phys Rep 486:75–174
12. Grabowicz P, Ramasco J, Moro E, Pujol J, Eguiluz V (2012) Social features of online networks: the strength of intermediary ties in online social media. PLoS ONE 7(1):e29358
13. Granovetter M (1973) The strength of weak ties. Am J Sociol 78(6):1360
14. Havemann F, Heinz M, Struck A, Gläser J (2010) Identification of overlapping communities by locally calculating community-changing resolution levels. arXiv:1008.1004
15. Karypis G, Kumar V (1999) Multilevel k-way hypergraph partitioning. In: Proceedings of the 36th annual ACM/IEEE design automation conference. ACM, New York
16. Kernighan B, Lin S (1970) An efficient heuristic procedure for partitioning graphs. Bell Syst Tech J 49(2):291–307
17. Kwak H, Lee C, Park H, Moon S (2010) What is Twitter, a social network or a news media? In: Proceedings of the 19th international conference on world wide web. ACM, New York, pp 591–600
18. Lancichinetti A, Fortunato S (2009) Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Phys Rev E 80(1):16118
19. Lancichinetti A, Radicchi F, Ramasco J, Fortunato S (2011) Finding statistically significant communities in networks. PLoS ONE 6(4):e18961
20. Lee C, Reid F, McDaid A, Hurley N (2010) Detecting highly overlapping community structure by greedy clique expansion. In: KDD SNA 2010
21. Leskovec J, Lang K, Dasgupta A, Mahoney M (2008) Statistical properties of community structure in large social and information networks. In: Proceeding of the 17th international conference on world wide web. ACM, New York, pp 695–704
22. Luce R, Perry A (1949) A method of matrix analysis of group structure. Psychometrika 14(2):95–116
23. McDaid A, Hurley N (2010) Detecting highly overlapping communities with model-based overlapping seed expansion. In: Advances in social networks analysis and mining (ASONAM), 2010 international conference on. IEEE Comput Soc, Los Alamitos, pp 112–119
24. Newman M (2004) Detecting community structure in networks. Eur Phys J B, Condens Matter Complex Systems 38(2):321–330
25. Palla G, Derényi I, Farkas I, Vicsek T (2005) Uncovering the overlapping community structure of complex networks in nature and society. Nature 435(7043):814–818
26. Reid F, McDaid A, Hurley N (2011) Partitioning breaks communities. In: 2011 Advances in social network analysis and mining
27. Shen H, Cheng X, Cai K, Hu M (2009) Detect overlapping and hierarchical community structure in networks. Phys A, Stat Mech Appl 388(8):1706–1712
28. Shi X, Adamic L, Strauss M (2007) Networks of strong ties. Phys A, Stat Mech Appl 378(1):33–47
29. Traud A, Kelsic E, Mucha P, Porter M (2011) Comparing community structure to characteristics in online collegiate social networks. SIAM Rev 53(3):526–543
30. van der Leij M, Goyal S (2006) Strong ties in a small world. Tinbergen Institute

31. Xie J, Kelley S, Szymanski B (2011) Overlapping community detection in networks: the state of the art and comparative study. arXiv:1110.5813
32. Yan B, Gregory S (2009) Detecting communities in networks by merging cliques. In: Intelligent computing and intelligent systems, 2009, ICIS 2009, IEEE international conference on, vol 1. IEEE Comput Soc, Los Alamitos, pp 832–836

SAINT: Supervised Actor Identification for Network Tuning

Michael Farrugia, Neil Hurley, and Aaron Quigley

Abstract Whenever the actors of a social network are not uniquely identifiable in the data, then entity resolution in the form of actor identification becomes a critical facet of a social network construction process. Here we develop SAINT, a pipeline for supervised entity resolution that uses relational information to improve, or tune, the quality of the constructed network. The first phase of SAINT uses attribute only based entity resolution to create an initial social network. Relational information between actors, actor network properties and other relational output of the first classification phase, are used in a second phase to improve the results of the original entity resolution. When compared to single phased approaches, the results from this two phased approach are consistently superior in both recall and precision measures. Embedded within SAINT are a series of evaluation checkpoints designed to measure both the quality of the individual classifiers and their impact within the entire pipeline. Our evaluation results provide insight on the potential propagation of error and open research questions for further improvement of the individual classifiers within the entire pipeline. As the main application of the process is to improve actor identification in social networks, we characterise the impact that entity resolution has on the final constructed network. We compare the network constructed using SAINT with a ground truth network using perfect entity resolution and use global and local network measures to study the differences.

Keywords Social networks · Social network construction · Actor identification · Entity resolution · Link prediction · Data mining

M. Farrugia (✉) · N. Hurley
University College Dublin, Dublin, Ireland
e-mail: mike.farrugia@gmail.com

N. Hurley
e-mail: neil.hurley@ucd.ie

A. Quigley
University of St Andrews, St Andrews, Scotland, UK
e-mail: aquigley@st-andrews.ac.uk

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_6,

1 Introduction

The data mining of social networks typically begins with an explicit social network collected by manual or automated means. Manual means include survey and interview, and automated means include data collection from social networking services such as LinkedIn with 100 m users or Facebook with 600 m users. In both cases, the collection process faces certain problems including data validity, terms of use, closed networks, privacy and security.

Researchers can also construct social network data sets from sources where the network is rather more implicitly defined. Email systems or a collected email corpus, such as the Enron Data set with 517,432 emails, can be analysed to determine a network. Here an email address becomes an actor in the network, an edge represents communication and a weight may represent the number of communications (direct, reply or CC) [31]. Likewise, personal geographic location data collected from mobile phones, can be used to infer and socially connect people living in the same area by matching on social events [32] or simple co-location. Alternatively, online documents can be matched to determine social networks using text classification [25]. Again, both the determination of actors and connections (edges) can be easily skewed by arbitrary judgements in the definition process of what constitutes an actor or connection.

More loosely defined social networks can be determined from the study of more open systems, including Twitter, blogs and other forms of social media. Here the notion of a "social connection" can be based on follower status, retweeting a tweet, family membership or even use of the same hashtag or URL in a blog post. Clearly, both manual, automatic and semi-automatic network formation methods all suffer from inherent weakness both in their means of data collection and the often subjective social concepts they are based around i.e. "what is a friend?" or "what is a social connection?" in this type of network.

The increasing uptake of online social networking services has opened up a range of business opportunities based on the successful extraction of social behaviours from such data. When the underlying social network is a customer base, it is clear that understanding the dynamics of this network in terms of how services are received and adopted by customers is important. It has the potential to allow businesses to, for example, better develop targeted services or organise marketing campaigns. The network of interest in this context, is one in which the nodes are the actors, or individuals about which the data has been gathered, and the edges, or links between actors, correspond to social relationships, such as friendship or family ties.

The relationships that are explicitly available in the raw data, tend to be recorded actions that link *references* to individuals. For example, in a dataset of email communication, an email is the action that links two references in the form of email addresses that relate to particular individuals. Now consider that a single social actor may use multiple email addresses and that the email communication may be evidence of a strong social link between a pair of actors, or just a spurious communication. It then becomes clear that the construction of a reliable representation of

the actual underlying social network, from the available raw data, presents a considerable challenge.

The airline passenger data used in this study offers a feature rich, real world example of an implicit network data set. The data set is built from airline booking travel records containing passenger details, flight itinerary details and passenger contact details. Often booking records do not contain details that allow a passenger to be uniquely identified and passenger name strings are ambiguous and inconsistent [10, 13]. Also, it is possible for the same passenger to have conflicting personal identification details in different travel records. For instance the same passenger can use a business email address when booking a business trip and a personal email address when travelling with their family. These characteristics of the data set make it a challenging problem both from an actor identification perspective and from a relationship inference perspective.

In this paper, we develop SAINT, an approach to social network construction, focusing in particular on the problem of actor identification, which is an instance of the *entity resolution* problem. Unlike other approaches in this area, we use relationships in the data records to assist in building an underlying social network. Our method consists of a pipeline of classifiers that infer an initial set of social actors and their relationships and then refine that set using the inferred social network. Our contribution in SAINT is to incorporate relational information into a supervised classification methodology. In a two-phase pipeline, a classical supervised attribute-based classification algorithm is used for initial actor identification and in the second phase a classifier based on feature vectors extracted from the inferred social network obtained in the first phase, is used to refine the identification. Crucially, our methodology provides several evaluation points at which the performance of the pipeline can be obtained, which allows for the tuning of the different classification phases. We evaluate our system using the traditional classification performance measures of precision and recall. Moreover, we compute various network properties to compare our extracted social network with the ground-truth network obtained using perfect actor identification. This allows the impact of classification errors on the resulting network to be investigated.

In the next section we present and review related work in entity resolution. Following this, Sect. 3 formally specifies the entity resolution problem that we are attempting to solve. Section 4 describes our entity resolution pipeline and details classification problems designed to improve the entity resolution process. Following this we describe a series of experiments on real world data in Sect. 5, along with proposed approaches for the previously described classification problems. Finally, we present and discuss results and motivate future work in Sect. 6.

2 Background

The problem of identifying multiple records referring to the same single entity has been studied since the seminal work of H.L. Dunn [14] and Fellegi and Sunter [18].

A common data mining approach is to apply machine learning classification on record attributes. Typically, the feature vectors that are input to the classifier correspond to pairs of records and the classifier output indicates whether or not this pair corresponds to the same underlying entity. This approach facilitates supervised entity resolution, where the classifier is trained on a set of pairs for which the resolution is known. More recently, the problem has been formulated as an unsupervised clustering problem in which records pertaining to the same entity are grouped together [6]. In this recent work relational information is exploited in the clustering algorithm.

Although the problem of entity resolution has been recognised for a long time, it is still considered as one of data mining's grand challenges [29]. In computer science, the same problem spans many different research communities, often under different names. The problem has been studied in databases as the merge/purge problem [19], natural language processing as coreference resolution [20], and computer vision as the correspondence problem [33]. In a social network context, entity resolution or actor identification is required when a unique key to identify the actors is not available, such as when constructing co-authorship networks. Authors in co-authorship data sets are not uniquely identified and sometimes an author's name is spelled differently, abbreviated, or there might be more than one author with the same name. There is little previous work which considers the problem purely from a social network perspective. A notable exception is the work of Newman [28], in which he creates two separate networks with different rules for determining the same authors. Bilgic et al. [7] describe a visual analytical system called D-dupe to assist with the process of actor identification in co-authorship networks.

The traditional entity resolution workflow is divided in five stages [12]. The principal steps in this workflow are; data cleaning, blocking to reduce the number of comparisons [4, 18], field comparison [13, 27], classification and evaluation [26]. Elmagarmid et al. [15] document the progress and advancements in entity resolution in a comprehensive survey paper which describes the state of the art throughout the various stages of entity resolution.

The traditional approach to entity resolution has been to use the record's attribute information to compare the similarity between the records. The same problem has been formulated in different ways [5] some of which exploit the nature of the data to infer more information than the attributes alone contain.

Availability of training data and advances in machine learning brought about the use of machine learning techniques to tackle entity resolution. The current state of the art in classification [11] uses Support Vector Machines (SVMs) for training models and classifying records, when training examples are available. SVMs have been successfully applied to several classification domains such as handwriting recognition, classifying facial expressions and text categorisation [9]. Originally, SVMs were designed to classify binary class problems, which makes them a prime candidate for entity resolution tasks, where the goal is to divide record pairs into two sets of matches and non-matches.

Recently relational information, such as child-parent relationships and co-authorship links between paper authors, has been used to improve the accuracy of

the classification process. The first relational entity resolution techniques treated relational information as another attribute in the comparison vector. These approaches still used a pairwise comparison process, but the additional relational information improves accuracy as coreferential entities are more likely to share relationships. In one such approach, Ananthakrishna et al. [2] describe a database centric approach that exploits data hierarchies in the database as additional relational information elements. This information is also used to reduce the number of comparisons in the entity resolution process.

Bhattacharya and Getoor [6] describe a more complete relational model with their collective entity resolution approach. They define entity resolution as a clustering problem where each cluster represents a unique entity. Clusters are merged based on their similarity which is calculated with a similarity measure that combines relational similarities and attribute similarities. The authors have shown that this approach improves both on attribute based entity resolution and on techniques that treat relationships as additional attributes.

3 Problem Formulation

In entity resolution, we are given a set of references $\mathscr{R} = \{r_i\}$ and the problem is to recover a set of hidden entities $\mathscr{E} = \{e_j\}$, such that each reference r corresponds to an entity in $\mathscr{E}$ and we write $e(r) \in \mathscr{E}$, as the entity corresponding to reference r and $C(e) \subseteq \mathscr{R}$ as the set of references mapped to the same entity e. In our application, entities are individuals or social actors in the social network and a reference is a data record pertaining to an individual. We use the terms node and entity interchangeably in this text due to the social network application domain. From the data, we can associate with each reference a set of attributes $A(r) = \{a_1, \ldots a_k\}$. Moreover, one can extract a *graph* over the references, $G(\mathscr{R}, L_{\mathscr{R}})$, where $L_{\mathscr{R}}$ is a set of *links* or edges that connect pairs of references. For example, a possible link may be that two references share a common attribute, such as a common phone number. Our approach to entity resolution involves the inference of a social graph $G(\mathscr{E}, L_{\mathscr{E}})$, consisting of links $L_{\mathscr{E}}$ connecting pairs of entities i.e. a network over the real social actors.

Following [6], we may view entity resolution as a clustering problem in which the goal is to group the references into the clusters that correspond to the same entity. Thus the set of references may be partitioned into $\mathscr{R} = C_1 \cup \ldots \cup C_k$, such that $C_i \cap C_j = \emptyset$ for $i \neq j$ and each cluster is associated with an entity in $\mathscr{E}$. Rather than clustering $\mathscr{R}$, it is more common to take a classification approach to entity resolution. Considering *pairs* of references $(r_i, r_j) \in \mathscr{R} \times \mathscr{R}$, a binary classifier can learn an output function which is 1 whenever $e(r_i) = e(r_j)$ and 0 otherwise. This approach facilitates supervised resolution, in which the binary classifier is trained given a set of training instances of corresponding reference pairs for which the resolution is known. Once binary classification is complete at the pairwise level, it is necessary to resolve the pairs into clusters to give the final entity resolution solution.

A common approach is to form the *transitive closure* of the classified pairs. That is, whenever $e(r_i) = e(r_j)$ and $e(r_j) = e(r_k)$ for a triple (r_i, r_j, r_k), then we infer $e(r_i) = e(r_k)$, through transitivity of the identity function. At the pairwise level, transitive closure involves the reclassification of negative pairs to a positive output, or the *merger* of these pairs into a single entity. Note that the pairwise classification can also contain inconsistencies (e.g. if the classifier decides $e(r_i) = e(r_j)$, $e(r_i) = e(r_k)$ but $e(r_k) \neq e(r_j)$ for some triple of references (r_i, r_j, r_k)), but transitive closure ignores such inconsistencies.

4 Entity Resolution Pipeline

In SAINT we take a two-phased approach to improve the quality of entity resolution. In the first phase we employ traditional supervised attributed based entity resolution. After this phase is complete, a social graph $G(\mathscr{E}, L_{\mathscr{E}})$ that links the resolved entities is constructed. In the second phase, the original data set is partitioned according to the output of the initial phase, with the initial negative instances processed differently to the positive instances. For the second phase we use two additional classifiers augmented with the new relational information from the inferred graph, to attempt to resolve errors introduced in the first phase. The result then represents a clear performance improvement in quality over the original output.

Errors in an entity resolution classification problem can take the form of either false negatives or false positives. False negatives translate into records that should be merged together but are not merged together. Conversely, false positives are records that should not be merged together but are merged together in entity resolution. In this paper we refer to the first type of errors (the false negatives) as *merge resolvable errors* and second type of errors (the false positives) as *split resolvable errors*. Menestrina et al. [26] refer to these types of errors as "broken errors" and "glued errors". For example, consider a set of references $\mathscr{R} = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}$ that need to be resolved. The set $\mathscr{T} = \{\{r_1, r_3, r_5\}, \{r_2, r_4\}, \{r_6, r_7, r_8\}\}$ is the ground truth and $\mathscr{E} = \{\{r_1\}, \{r_3, r_5\}, \{r_2, r_4, r_6, r_7, r_8\}\}$ the output of the first entity resolution stage. In $\mathscr{E}$ clusters $\{r_1\}$ and $\{r_3, r_5\}$ are an example of a merge resolvable error and need to be merged into a single set. The set $\{r_2, r_4, r_6, r_7, r_8\}$ is a split resolvable error and needs to be split into two separate sets. It is clear that some situations may require a combination of split and merge operations in order to attain the correct ground truth from the initial partitioning.

Figure 1 illustrates the complete pipeline showing the interaction between the different phases. The impact of the whole process is made transparent from an evaluation standpoint by measuring the performance at various points in the pipeline and using these measurements to fine tune the process. In the diagram each circle is an evaluation checkpoint and the letters next to the checkpoints are used for reference when reporting the results. The merge classifier and the split classifier can be applied in sequence, for instance first applying split then merge, or vice versa. Alternatively the input of all the 4 classification stages in this process can be ensembled to merge the results together.

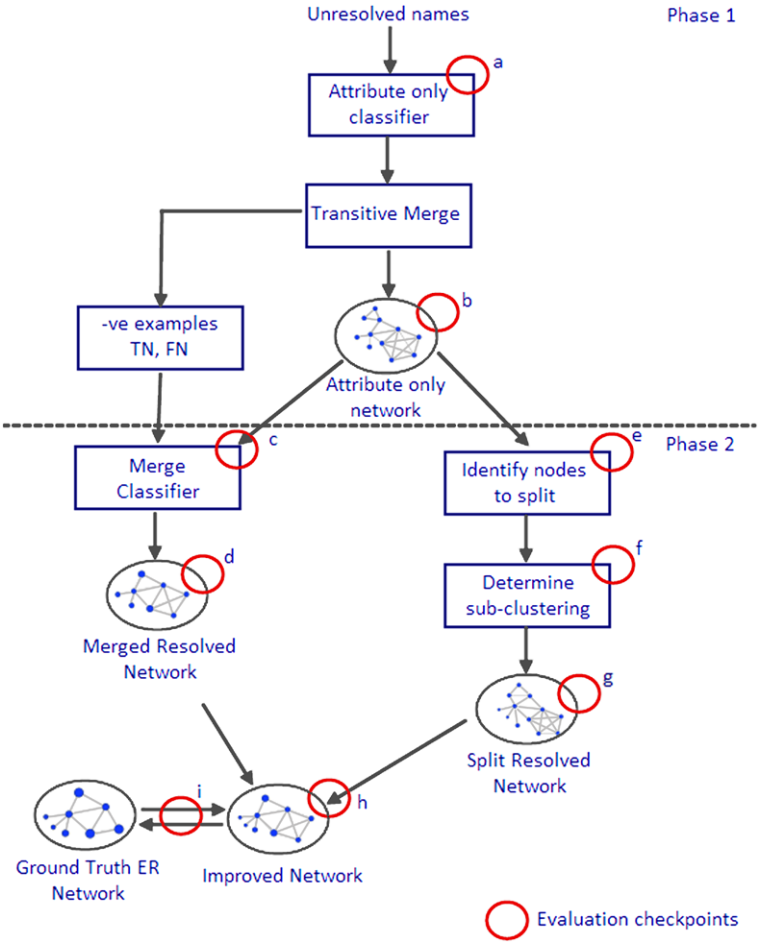


Fig. 1 Entity resolution pipeline

4.1 Merge Classifier Problem Definition

The merge classifier is applied to all negative instances from the initial attribute-based classifier and its purpose is to distinguish the false-negative outputs of the initial phase from among all its negative outputs. When evaluating one does not know which instances are true negative and which are false negatives, therefore all negative instances have to be considered. By looking only at the negative examples, the number of records that are considered at this stage is reduced over the initial phase. Entity resolution is a typical example of a classification task that tends to suffer from the class imbalance problem [34], leading to the majority of records being negative instances and only a few positives. Since often the negative instances

constitute the majority of record comparisons, if scalability is an important concern, it is useful to look at possibilities to reduce the number of comparisons at this stage. Since the very first approaches to entity resolution, starting with the Fellegi-Sunter model, classifiers have split the resolution output into three classes; matches, non-matches and a third class of possible matches that were intended for manual review. If the first phase of entity resolution uses an approach that either produces a 'possible matches' set, or alternatively outputs a confidence value on the accuracy of a match, then only these items can be considered during this second phase. Alternatively, the importance of nodes to review in this phase can be calculated based on the *graph impact*, by selecting only those merge operations that would impact mostly on the structure of the final network. The output of the merge classifier is a list of pairs of nodes from the original network that should be merged together.

4.2 Split Classifier Problem Definition

Each node of the network formed after the first phase of classification corresponds to a cluster of references, $C \subseteq \mathscr{R}$, that were combined into a single entity by the attribute-based classifier and transitive closure. The split classifier consists of two tasks. The first task is to identify candidate nodes that may contain references that should be split. The second task is to apply a clustering or component identification process to the candidate nodes to further refine its corresponding cluster of references into sub-clusters i.e. to find a partitioning of $C = C_1 \cup \cdots \cup C_l$, thus splitting the nodes into l separate entities for some appropriate number of clusters l.

Similarly to the merge classifier, a classification approach can be applied for identification of candidate nodes, where the positive outputs of the first phase *including* the transitive merge step, are input into a second classifier that tries to identify the false positives from amongst these. In fact, from empirical tests, we find that the nodes that need to be split are often the result of a naive transitive merge step. The split classifier is essentially applying a more refined approach to undo the coarse initial transitive merger where appropriate. We achieve this refinement using network information. In particular, consider the graph $G(\mathscr{R}, L_{\mathscr{R}})$ formed over the references where a link exists between a pair of references if the attribute-based classifier of phase one has determined that these references map to the same entity. Transitive closure is equivalent to identifying the connected components of this graph as single entities. Instead, we build a feature vector to represent each component, consisting of attributes of its constituent references, as well as graph properties of the connected component. A classifier is trained over the resulting feature vectors to determine the candidate nodes that need to be split. Finally, a clustering algorithm is applied to each candidate node to determine the sub-clustering.

Table 1 SAINT: Pipeline process overview

The steps of the process are the following:
Phase 1:
(a) Apply 'traditional' attribution based ER (b) Apply transitive closure on the pairwise output (c) Generate a network based on the initial ER results
Phase 2:
(d) Merge classifier: Using data from the extracted network, reclassify all negative instances to identify missed merged nodes (e) Split classifier: (i) Identify nodes that were incorrectly merged based on their internal network and attribute structure (ii) Split these nodes using a combination of community finding graph classification and attributes
Evaluation:
(a) Each stage and each classifier can be evaluated separately and in conjunction with the rest of the ER pipeline. (b) The initial ER network and the improved network can be compared with the ground truth network with perfect ER to characterise the network difference.

5 Method

5.1 Data

Before describing in detail candidate solutions for the problems described in the previous section, it is useful to discuss the data set that was employed in this research. The data is extracted from a real life airline passenger database that contains all the booking data for passengers travelling with one individual airline. For this experiment the data can be used in this research as long as it remains held confidential and unidentified. The booking data contains at a minimum the 'name' of the passengers, the travel itinerary and the sale origin (e.g. web, travel agent etc). Additionally bookings typically contain passenger contact information such as phone number, email address and home address. In the general case, there is not an easy way to uniquely and unambiguously identify an airline customer based on the information contained in the booking alone. Passenger names are ambiguous and contact information cannot be naively used to identify customers. The only uniquely distinguishing element in such data sets is a loyalty card number (credit card information and passport information is not available for security and confidentiality reasons).

Airline loyalty numbers in the form of frequent flyer numbers provide a viable training set and ground truth for training and testing algorithms. Depending on the popularity of a particular airline's frequent flyer program, between 7 %–30 % of the passengers have a frequent flyer number. The application of this research is to

uniquely identify passengers travelling multiple times, even when they are not members of a frequent flyer program. This process effectively transforms simple passenger numbers into a valuable business resource in the form of customer data [16]. This is possible by first training algorithms based on the known frequent flyers and then applying the output on non frequent flyers. The data available for frequent flyers is not different or biased from that of non-frequent flyers.

5.2 Attribute Classifier

For the attribute classifier 3 classification techniques traditional used for entity resolution are compared. The three methods include; a supervised SVM classifier, the Fellegi-Sunter approach and a manually constructed rule based classifier, developed based on domain expertise. Each classifier uses the same input vector that compares the attribute similarity between the two full passenger records. These include similarity between passenger names based on the Jaro-Winkler string measure [30] and comparison between the other passenger information such as email, phone, address and travel itinerary similarity.

Transitive closure is applied to the output of the pairwise attribute classifier. Consistent with the findings in [11], the best classifier from those tested is the supervised classifier, in this case an SVM classifier. The rest of the processes in the pipeline are applied using the best attribute classifier, to attempt to improve the quality of the output.

5.3 Creating the Network

After the first stage of entity recognition in SAINT, a social network is created to link passengers together. Each node in the network is a uniquely identified passenger output from the first attribute classifier. The edges are defined by a sequence of rules as follows:

1. PNR: passengers who are booked on the same booking record,
2. MULTI: passengers who have travelled on the same flight multiple times,
3. ADDRESS: passengers who share the same home address,
4. PHONE: passengers who share the same phone number,
5. EMAIL: passengers who share the same email,
6. DOMAIN: passengers who share the same email domain address (filtered out for common domain address such as common mail providers and internet service providers).

The edges are weighted according to the respective importance of the links based on previous studies [17] that manually collected passenger data to understand the respective link importance. The final network consists of a combination of all the edges created from these rules. This generated network is the input to the second

phase of the pipeline. This network is also the baseline comparison to quantify the improvement in the second phase of the entity resolution pipeline.

5.4 Merge Classifier

The purpose of the merge classifier is to identify nodes that should have been merged during entity recognition but were not merged, most likely because the matching attributes were either not available or included conflicting information. It is not uncommon for passengers to use, say, different email addresses or phone numbers, particularly if they distinguish between business and leisure travel.

The intuition to use network information for this process is that identical nodes that were not merged, are likely to travel with the same people, therefore share common neighbours in the network. While in the case of different purposes for travel, overlapping neighbours might not be the same, the network distance between two identical passengers is still likely to be small.

One can also consider the merge classifier as a link prediction problem, where each link between two nodes means that the nodes are identical. In fact, the list of network measures used for this classifier was inspired by the work of [22] and [23]. Due to the large number of comparisons that need to be made, preference was given to fast neighbourhood based measures to maintain scalability. Measures such as the Katz measure [21] that were reported to be highly predictive in previous research were considered separately due to their increased time complexity.

The final input vector used for the merge classifier included, amongst other basic network measures; the number of common neighbours, Jacard similarity, Adamic-Adar measure [1], and preferential attachment measure. We also incorporate path length information in the form of basic path length, weighted path length and in a separate example the Katz measure (using $e = 0.05$ with cutoff point at 4 degrees). The network measures where selected amongst the list of measures in literature [22, 23] based on the highest information gain. We also include information of the types of edges present between the two nodes based on the types of edges between the nodes as described in Sect. 5.3. We also add two attributes measures between the nodes; the name and surname Jaro-Winkler similarity measures.

For the merge classifier we employ 2 types of supervised classifiers, SVMs and random forests. One of our primary interests is to understand the additional impact of relational information to improve the entity resolution result. We therefore compare models using relational measures with models that include only the two attribute measures. We also conduct a separate comparison using expensive network metrics, such as the Katz measure.

5.5 Identifying Nodes to Be Split

Entities that should be split are ones which contain multiple entities in the same cluster. The aim is therefore to identify the dirty clusters that need to be split. In view

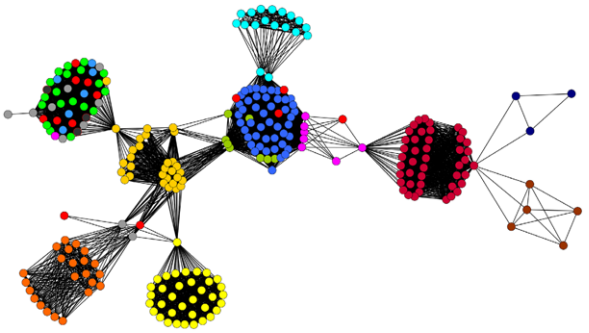


Fig. 2 Splitting a node

of this, the vector information consists of measures that describe the references that make up a cluster and the uniqueness of these references. For instance, one attribute measurement is encoded by the proportion of unique names and surnames within a node. Another measure is the global name and surname frequency measures over the whole data set, since often initial misclassifications were related to the most common names.

The relational information for split identification includes information on the internal network structure of the node. This included the number of nodes and edges, the average clustering coefficient, the density of the internal networks and the node degree centrality of the particular node within the global network.

Two supervised classifiers, an SVM and a random forest classifier, are used to identify the nodes that should be split. Each classifier was run twice, once with the attribute only vector and once with the attribute and relational information.

5.6 Determine Sub-clustering

As discussed in Sect. 4.2, the internal network perspective of each entity provides valuable insight on a potential splitting strategy for nodes that have been misclassified. Figure 2 illustrates a real example of a node that was misclassified in the first step. The colours of each node in the image represent the different individuals that were incorrectly merged together into one node. As one can notice the internal structure of the node and the organisation provides a good indication of how the node should be split.

This problem can be framed as either a community finding problem where each community is split into a different entities, or else it can also be seen as a graph classification problem. We consider 4 approaches to split nodes. The first approach is a simple attribute only based approach that splits nodes internally on the unique name and surname of each person, i.e. every record with the same unique name and surname combination is grouped together. This does not include any relational information but provides a good baseline to compare against. The second approach is to use a simple relational neighbour classifier described by Macskassy and Provost [24]

based on the neighbourhood of each node to label each node. The third approach uses the Blondel fast community finding algorithm [8] based on modularity optimisation to split the node, thus using only information related to the relational organisation of the node. The final approach uses the collective graph clustering approach described by Bhattacharya and Getoor [6].

6 Evaluation

6.1 Data

For the evaluation of SAINT in this study we extract 4 sample networks of different sizes from the main 20 million record airline data set. The initial network sampling seed includes all bookings that contain one randomly selected email address domain. Noise from the other passenger records in the original data was added to this sample seed, to ensure that enough duplicate names existed in the data to ensure that the solution is not trivial. To continue sampling the network, the neighbours of the selected sample are then added to the network up to the 7th iteration considering only the PNR relation, which is the strongest relation. The sizes of the networks are outlined in Table 2. The table reports the total number of passenger records (references), the number of unique entities, the ratio between the two, the number of unique names and surnames and the proportion of unique names and surnames of the total number of records.

Sample A was considered as the network model training data set and networks B, C and D are the test sets. The reported results on network A are conducting using 10-fold cross validation.

6.2 Results

To evaluate the improvement of the proposed approach we use the traditional information retrieval measure of precision, recall and a combined f-measure score [3]. The accuracy measure is omitted due to the predominance of true negative records in the classification task that makes this measure less relevant. While the f-measure metric attempts to combine the values of precision and recall into one measure, we report precision and recall separately, to fully understand the impact that the different improvement processes have on the quality performance. As entity resolution is ultimately a clustering task, clustering measures, such as, variation of information, were considered and reported in the final summary result in Table 7. We report this result to illustrate the significance of improvement from a clustering perspective.

There is an inherent tradeoff between precision and recall. In a social network context, sometimes a more precise network can be more important than a more complete network and vice-versa. For example, if one is interested in the most central

Table 2 Datasets size and properties

Network	Recs	Unq	Ratio	Names	Ratio
A	54,010	20,277	2.664	23,251	0.430
B	38,227	13,411	2.850	16,138	0.422
C	94,852	24,447	3.880	25,890	0.273
D	53,949	16,989	3.176	22,445	0.416

nodes in a network and the network has a high recall but low precision, effectively having more than one real entity merged together in a single entity, then the most central nodes might be reported incorrectly because they are made up of a combination of several real entities. Our approach in SAINT empowers the researcher to tune the resultant network and improve the network in the required direction.

Tables 3–6 show the evaluation of the entity resolution process pipeline throughout the most critical stages. In each table the precision, recall and f-measure for each respective classification approach is reported for the 4 different data sets (A–D). Table 3 reports the output of the pairwise original classification result. This corresponds to evaluation step A in Fig. 1. Table 4 shows the results after applying transitive closure. This corresponds to step B in Fig. 1. Tables 5 and 6 report the result of the merge and split classifiers applied to the best result obtained in evaluation step B, i.e. the SVM model highlighted in Table 4. These two evaluation steps correspond to steps D and G in Fig. 1.

Table 7 shows a summary of the improvement in the results from the original attribute classifier, the two additional improvement classifiers and the final combination of the result. In this case the results were combined by first applying the merge classifier then applying the split classifier on this output. We also tried to apply the classifier in the reverse order with a very similar result.

As the focus of this work is predominantly on entity resolution for network construction, it is important to consider the impact that errors in the actor identification stage of network construction can have on the constructed network. Intuitively, networks with a high degree of merge errors tend to be sparser than the respective ground truth network, because more nodes are present in the constructed network. Conversely, split errors may lead to the creation of hubs which do not really exist. From our tests we found no significant difference in the global network measures between the different networks. We considered the measures of; number of nodes and edges, size of the giant connected component, network density, diameter of the giant connected component, average shortest path and average node degree.

To understand the impact on a local individual node level, the highest ranking nodes based on degree centrality and betweenness centrality were calculated. Figure 3 shows the result of the degree centrality between the ground truth, original network and the final improved network from network A. Cells with the same colour refer to identical passengers in each network, whereas uncoloured cells are cells that do not match between the two lists. The node names are abbreviated by initials to retain anonymity. Upon inspection we found that a number of high ranking hubs

Table 3 Attribute classifier (pair wise result) – evaluation step A

Model	Dataset A			Dataset B			Dataset C			Dataset D		
	Pre	Rec	F-m	Pre	Rec	F-m	Pre	Rec	F-m	Pre	Rec	F-m
SVM	0.9628	0.8318	0.8925	0.9431	0.8418	0.8896	0.8937	0.7986	0.8435	0.9713	0.7861	0.8690
Fellegi-Sunter	0.9494	0.6820	0.7938	0.9355	0.6912	0.7950	0.7769	0.6906	0.7312	0.9351	0.6692	0.7801
Rule Base	0.9701	0.6973	0.8114	0.9743	0.7031	0.8168	0.9028	0.7080	0.7936	0.9725	0.6947	0.8104

Table 4 Attribute classifier (transitive merge) – evaluation step B

Model	Dataset A			Dataset B			Dataset C			Dataset D		
	Pre	Rec	F-m	Pre	Rec	F-m	Pre	Rec	F-m	Pre	Rec	F-m
SVM	0.7818	0.9484	0.8571	0.8105	0.9446	0.8724	0.2940	0.8931	0.4424	0.7495	0.9455	0.8362
Fellegi-Sunter	0.6798	0.7681	0.7213	0.6735	0.7747	0.7205	0.2198	0.7783	0.3428	0.6087	0.7924	0.6885
Rule Base	0.8357	0.7668	0.7998	0.8736	0.7764	0.8221	0.3457	0.7776	0.4786	0.8213	0.8293	0.8253

Table 5 Merge classifier – evaluation step D

Model	Dataset A			Dataset B			Dataset C			Dataset D		
	Pre	Rec	F-m	Pre	Rec	F-m	Pre	Rec	F-m	Pre	Rec	F-m
SVM Relational	0.7811	0.9529	0.8585	0.8107	0.9502	0.8749	0.2942	0.8946	0.4428	0.7499	0.9502	0.8382
Forest Rel	0.7853	0.9702	0.8680	0.8127	0.9609	0.8806	0.2970	0.9060	0.4473	0.7532	0.9669	0.8468
Forest Attr Only	0.7821	0.9516	0.8585	0.8108	0.9481	0.8741	0.2949	0.8969	0.4439	0.7501	0.9487	0.8378
Forest Rel + Katz	0.7855	0.9893	0.8757	0.8122	0.9592	0.8796	0.2970	0.9061	0.4474	0.7527	0.9643	0.8455

Table 6 Split classifier – evaluation step G

Model	Dataset A			Dataset B			Dataset C			Dataset D		
	Pre	Rec	F-m	Pre	Rec	F-m	Pre	Rec	F-m	Pre	Rec	F-m
Name split	0.9471	0.8767	0.9105	0.9199	0.9265	0.9232	0.7783	0.8529	0.8139	0.9548	0.9285	0.9415
Collective Inf	0.9601	0.9434	0.9517	0.9320	0.9409	0.9364	0.9034	0.8850	0.8941	0.9668	0.9423	0.9544
Clustering Split	0.9568	0.9028	0.9290	0.9264	0.9379	0.9321	0.7616	0.8728	0.8134	0.9416	0.9367	0.9391
Coll. Graph Clust.	0.9924	0.8504	0.9159	0.9539	0.9173	0.9353	0.9778	0.8172	0.8903	0.9824	0.9242	0.9524

Table 7 Summary of results with reported improvement difference between classifiers

Model	Dataset A							Dataset B						
	Pre	Diff	Rec	Diff	F-m	Diff	VI	Pre	Diff	Rec	Diff	F-m	Diff	VI
Original	0.782		0.948		0.857		0.115	0.810		0.945		0.872		0.111
Merged	0.785	0.003	0.970	0.022	0.868	0.011	0.097	0.813	0.002	0.961	0.016	0.881	0.008	0.097
Split	0.960	0.178	0.943	−0.005	0.952	0.095	0.077	0.932	0.121	0.941	−0.004	0.936	0.064	0.085
Combined	0.960	0.178	0.965	0.017	0.963	0.106	0.058	0.930	0.120	0.957	0.013	0.943	0.071	0.072

Model	Dataset C							Dataset D						
	Pre	Diff	Rec	Diff	F-m	Diff	VI	Pre	Diff	Rec	Diff	F-m	Diff	VI
Original	0.294		0.893		0.442		0.240	0.749		0.946		0.836		0.102
Merged	0.297	0.003	0.906	0.013	0.447	0.005	0.228	0.753	0.004	0.967	0.021	0.847	0.011	0.084
Split	0.903	0.609	0.885	−0.008	0.894	0.452	0.128	0.967	0.217	0.942	−0.003	0.954	0.118	0.075
Combined	0.904	0.610	0.898	0.005	0.901	0.459	0.116	0.963	0.214	0.964	0.018	0.964	0.127	0.058

	NETWORK A - DEGREE CENTRALITY		
Rank	Ground Truth	Original Network	Improved Network
1	J T (0.03379)	M S (0.03745)	N T (0.03520)
2	M T (0.03363)	M A S (0.03638)	J T (0.03520)
3	N T (0.03363)	N T (0.03476)	M T (0.03514)
4	C T (0.03357)	M T (0.03471)	C T (0.03498)
5	J A (0.03312)	J T (0.03471)	J A (0.03434)
6	M L (0.03255)	C T (0.03454)	M D (0.03370)
7	M D (0.03227)	J A (0.03390)	M L (0.03365)
8	K H (0.03221)	R W (0.03374)	R G (0.03360)
9	J S (0.03210)	J B (0.03363)	J B (0.03354)
10	R G (0.03193)	D G (0.03363)	J H (0.03349)
11	J B (0.03176)	M S 2 (0.03352)	M H (0.03344)
12	K L (0.03165)	J S 2 (0.03347)	K H (0.03344)

Fig. 3 Degree centrality variation between the networks

were incorrectly identified in the original network but were corrected in the improved network. For example, the first two passengers in the original network were incorrectly resolved entities that had multiple (up to 20) unique names identified as a single entity.

7 Discussion

At the heart of SAINT is a two phased approach to solving the entity resolution problem. The approach starts by taking the output of a state of the art classifier that is often used in entity resolution, and by adding relational information from a intermediate constructed network, improves the quality of both the entity resolution and the constructed network. Our results show that the approach developed in SAINT consistently improves the quality of the original state of the art classifier used on the same data. Unlike other approaches that use relational information in entity resolution, SAINT describes a method for first deriving the relational information and then using this information for supervised classification.

A key observation in the process is the impact of applying transitive closure on the quality of the results. Recall increases dramatically after applying transitive closure, heavily sacrificing the precision. In the worst case precision drops by around 59 %, due to the particular network sample containing low demographic information about the passengers (i.e. phone, email, address details). Thus the attribute-based classifier was able to make correct classifications linking the relatively few records for which information was available but also created effectively random edges joining these high-quality components into larger connected components. Transitive closure then resulted in the creation of a large number of new positives, most of

which were false. This provides evidence that naive transitive closure will deteriorate the overall classification in many cases and that a more refined method for merging pairwise decisions into clusters is required. In effect, our split classifier provides one means for which this can be achieved.

In general the merge classifier that was intended to improve recall did not improve recall by a significant amount. There was a small improvement ranging between 1 % and 2 % across the 4 data sets. Despite this, the recall was already in the 90 % range before the application of the merge classifier. The difference between the 4 different classification approaches was also very slight, within 2 %. While the 3 classifiers that included relational information in general performed better than the one with attribute information only, the difference was sometimes marginal. One would argue that the importance of the merge classifier might need to be evaluated better in a scenario where recall is not already so high, in order to assess the true impact of the classifier.

Conversely, the split classifier had a high quality improvement on precision and therefore the general quality of the resultant network. This impact is mostly pronounced in network C were previously transitive closure heavily degraded the quality of the network. From the four attempted classifiers collective inference provided the best improvement, although the other approaches also performed well. Unsurprisingly due to the similarity between the two approaches, the collective graph clustering approach came close second. Both the relational only classifier (clustering split), and the attribute only classifier (name split), produced adequate results, with the relational only classifier being slightly superior.

When performing entity resolution for the purpose of network construction the overall goal is to derive a network that is as close to the ground truth network as possible. We expected certain differences in terms of global network measures between the different stages in the network yet the difference between the measures was insignificant. Perhaps one reason for this is that within the size of the networks considered, the impact of smaller errors in entity resolution is lost within the context of the network. On the other hand we noticed that entity resolution did effect the identified most central nodes which is relevant in most social networking studies. In this regard, one can attempt to incorporate the impact of entity resolution on the social network in the entity resolution process itself. For instance, merging peripheral nodes in the network might not be as significant as merging or splitting the more central nodes. Similarly, adding an edge to a network can have a huge impact on the measures and the structure of the network, depending on where the edge is added.

This leads to the notion of the *graph impact* that a merge or split operation can have on the network. The graph impact can be a measure of network change that an operation has on the network, depending on the network research question. For instance, the impact of a split or merge operation that results in two components being merged or split, can be quantified to have an impact proportional to the size of the two components. When training models to recognise the split and merge operations one can also weight the training instances so that operations that have a higher graph impact are given more importance in terms of both precision and recall. Extending this, graph impact measures can be part of the model attribute

information. Similarly, when evaluating the results, decisions with high impact can be weighted to have a higher significance than operations that have marginal impact on the network.

8 Conclusion

In this paper we show how the quality of the traditional entity resolution process can be improved using relational information that is generated from the first step of the SAINT process. We tailor our study particularly towards network construction to focus on the most important aspects related to this problem. This also influences the resultant evaluation of the entity resolution process and the impact it has on the network. Future work will consider the inclusion of the notion of graph impact within the entity resolution process, the application of this proposed model on the original large scale data set of around 20 million records and the possibility of developing an iterative step using multiple instances of merge and split classifiers and ensembling their output together.

References

1. Adamic L, Adar E (2003) Friends and neighbors on the web. Soc Netw 25:211–230
2. Ananthakrishna R, Chaudhuri S, Ganti V (2002) Eliminating fuzzy duplicates in data warehouses. In: Proceedings of the 28th international conference on very large data bases, VLDB'02, VLDB Endowment, pp 586–597
3. Baeza-Yates R, Ribeiro-Neto B (1999) Modern information retrieval. ACM, New York
4. Baxter R, Christen P, Churches T (2003) A comparison of fast blocking methods for record linkage. In: Proceedings of the KDD-2003 workshop on data cleaning, record linkage, and object consolidation, Washington DC, vol 3. pp 25–27
5. Benjelloun O, Garcia-Molina H, Kawai H, Larson TE, Menestrina D, Su Q, Thavisomboon S, Widom J (2006) Generic entity resolution in the serf project. Technical Report 2006-14, Stanford InfoLab
6. Bhattacharya I, Getoor L (2007) Collective entity resolution in relational data. ACM Trans Knowl Discov Data 1:5
7. Bilgic M, Licamele L, Getoor L, Shneiderman B (2006) D-dupe: an interactive tool for entity resolution in social networks, 31 2006–Nov. 2, pp. 43–50.
8. Blondel V, Guillaume J, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008:P10008
9. Burges C (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Discov 2:121–167
10. Christen P (2006) A comparison of personal name matching: techniques and practical issues. Tech. Rep. TR-CS-06-02
11. Christen P (2008) Automatic record linkage using seeded nearest neighbour and support vector machine classification. In: KDD '08: proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, pp 151–159
12. Christen P, Churches T, Hegland M (2004) Febrl – a parallel open source data linkage system. In: Dai H, Srikant R, Zhang C (eds) Advances in knowledge discovery and data mining. Lecture notes in computer science, vol 3056. Springer, Berlin, pp 638–647

13. Cohen WW, Ravikumar P, Fienberg SE (2003) A comparison of string distance metrics for name-matching tasks, pp 73–78
14. Dunn H (1946) Record linkage. Am J Publ Health 36:1412
15. Elmagarmid AK, Ipeirotis PG, Verykios VS (2007) Duplicate record detection: a survey. IEEE Trans Knowl Data Eng 19:1–16
16. Farrugia M, Quigley A (2009) Enhancing airline customer relationship management data by inferring ties between passengers. In: Proceedings of the international conference on social computing
17. Farrugia M, Hurley N, Quigley A (2011) Snap: towards a validation of the social network assembly pipeline. In: International conference on advances in social network analysis and mining, pp 228–235
18. Fellegi I, Sunter A (1969) A theory for record linkage. J Am Stat Assoc 64:1183–1210
19. Hernández M, Stolfo S (1998) Real-world data is dirty: data cleansing and the merge/purge problem. Data Min Knowl Discov 2:9–37
20. Hirschman L, Chinchor N (1997) Muc-7 coreference task definition – version 3.0
21. Katz L (1953) A new status index derived from sociometric analysis. Psychometrika 18:39–43
22. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58:1019–1031
23. Lü L, Zhou T (2011) Link prediction in complex networks: a survey. Phys A, Stat Mech Appl 390(6):1150–1170
24. Macskassy S, Provost F (2003) A simple relational classifier. In: Proc. of the 2nd workshop on multi-relational data mining (MRDM 03), pp 64–76
25. Makrehchi M, Kamel M (2007) A text classification framework with a local feature ranking for learning social networks. In: 2007 seventh IEEE international conference on data mining, ICDM 2007, pp 589–594
26. Menestrina D, Whang S, Garcia-Molina H (2010) Evaluating entity resolution results. Proc VLDB Endow 3:208–219
27. Navarro G (2001) A guided tour to approximate string matching. ACM Comput Surv 33:31–88
28. Newman M (2001) Scientific collaboration networks. I. Network construction and fundamental results. Phys Rev E 64:16131
29. Piatetsky-Shapiro G, Djeraba C, Getoor L, Grossman R, Feldman R, Zaki M (2006) What are the grand challenges for data mining, KDD-2006 panel report. ACM SIGKDD Explor Newsl 8:70–77
30. Porter E, Winkler W, of the Census B, States U, Division SR (1997) Approximate string comparison and its effects on an advanced record linkage system. US Bureau of the Census
31. Qiu J, Lin Z, Tang C, Qiao S (2009) Discovering organizational structure in dynamic social network. In: 2009 ninth IEEE International conference on data mining, ICDM '09, pp 932–937
32. Quercia D, Lathia N, Calabrese F, Di Lorenzo G, Crowcroft J (2010) Recommending social events from mobile phone location data. In: 2010 IEEE 10th international conference on data mining, ICDM, pp 971–976
33. Scharstein D, Szeliski R (2002) A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int J Comput Vis 47:7–42. Has 1205 citations
34. Tan P, Steinbach M, Kumar V (2005) Introduction to data mining. Addison-Wesley, Reading

Holder and Topic Based Analysis of Emotions on Blog Texts: A Case Study for Bengali

Dipankar Das and Sivaji Bandyopadhyay

Abstract The paper presents an extended approach of analyzing emotions of the blog users on different topics. The rule based techniques to identify emotion *holders* and *topics* with respect to their corresponding *emotional expressions* helps to develop the baseline system. On the other hand, the Support Vector Machine (SVM) based supervised framework identifies the *holders*, *topics* and *emotional expressions* from the blog sentences by outperforming the baseline system. The existence of many to many relations between the *holders* and the *topics* with respect to Ekman's six different emotion classes has been examined using two way evaluation techniques, one is with respect to *holder* and other is from the perspective of *topic*. The results of the system were found satisfactory in comparison with the agreement of the subjective annotation. The error analysis shows that the *topic* of a blog at document level is not always conveyed at the sentence level. Moreover, the difficulty in identifying *topic* from a blog document is due to the problem of identifying some features like bigrams, Named Entities and sentiment. Thus, we employed a semantic clustering approach along with these features to identify the similarity between document level topic and sentential *topic* as well as to improve the results of identifying the document level topic.

Keywords Emotion · Blog · Holder · Topic · SVM · NE · Discourse relation

1 Introduction

Emotion analysis is a recent sub discipline at the crossroads of information retrieval [1] and computational linguistics [2]. A wide range of Natural Language Processing (NLP) tasks such as classification of newspaper articles [3], blogs [4], question answering systems [5], modern information retrieval systems [6] are using

D. Das (✉) · S. Bandyopadhyay
Department of Computer Science and Engineering, Jadavpur University, Kolkata, India
e-mail: dipankar.dipnil2005@gmail.com

S. Bandyopadhyay
e-mail: sivaji_cse_ju@yahoo.com

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_7,

emotional information. The research activities in the areas of emotion in natural language texts and other media are gaining ground under the umbrella of subjectivity analysis and affect computing. The reason may be the explosive growth of the social media content on the Web in the past few years.

Emails, weblogs, chat rooms, online forums and even twitters are being considered as the affective communication substrates to analyze the reaction of emotional catalysts. Blog is an important communicative and informative repository of text based emotional contents in the Web 2.0 [4]. Especially, the blog posts contain instant views, updated views or influenced views regarding single or multiple topics. Many blogs act as an online diary of the bloggers for reporting the blogger's daily activities and surroundings. Sometimes, the blog posts are annotated by other bloggers. On the other hand, large collection of blog data is suitable for any machine learning framework. But, the identification of emotions expressed on a *topic* in the text with respect to a *reader* or *writer* is itself a challenging task [7]. Recently, the identification of the temporal trends of sentiments and topics has also drawn attention of NLP communities [8]. The tracking of emotions over events [9] or about politics as expressed in online forums or news to customer relationship management, the determination of emotion *holder* and *topic* is an important task. Extraction of emotion *holder* is important to discriminate between emotions that are viewed from different perspectives [10]. Among all concerns, emotions of people are important because people's emotions have great influence on our society. By grouping emotion *holders* of different stance on diverse social and political issues, we can have a better understanding of the relationships among countries or among organizations [11]. Thus, the perspectives of sociology, psychology and commerce along with the close association among people, topic and emotions motivate us to investigate the insides of emotional changes of people over topic and time.

The present task deals with the identification and analysis of users' emotion on different *topics* from an annotated Bengali blog corpus [12]. A simple rule based baseline system is developed to identify the *emotional expressions*, *holders* and *topics*. The expressions are identified from shallow parsed sentences using Bengali WordNet Affect [13] whereas a simple part-of-speech (POS) based pattern matching technique is employed to identify the emotion *holders* and *topics* with respect to the *emotional expressions*. On the other hand, the Support Vector Machine (SVM) [14] based supervised classifier is employed and the required feature vectors are prepared based on the clues present in the sentences such as lexical, syntactic, semantic, rhetoric and overlapping (word, part-of speech (POS), Named Entity (NE)). The supervised system outperforms the baseline system by achieving the F-scores of 72.54 %, 61.02 % and 57.09 % for the *emotional expression*, *holder* and *topic*, respectively, on 512 test sentences.

Topic identification is essential in connection within categorizing search applications [15]. The categorizing search has attracted much interest recently; its potential has been realized by users and search engine developers in the same way. Categorizing search means to apply text categorization facilities to retrieval tasks where a large number of documents is returned. Consider for example the use of Internet search engines like Google or Lycos: Given a query they deliver a bulky result list

D of documents. Categorizing search means to return D as a set of priori unknown categories such that thematically similar documents are grouped together.

But, in case of blog documents, it is observed that the *topics* discussed by the bloggers in their comment sections especially at sentence level are not similar with the *topic* of the blog document on which they are commenting. Sometimes, the sentential *topics* in the comment sections of the bloggers are completely irrelevant with the *topic* of the overall document. Thus, it is necessary to distinguishingly determine the sentential *topics* as well as the document topics. Thus, it is important to find methods that can annotate and organize blog documents in meaningful ways so that the topic identification can also be used for document ranking in Informational Retrieval systems. In addition to the content of the blog document itself, other relevant information about a document such as related comments with sentential *topics* can often enable a faster and more effective search or classification. Hence, in the present task, we have used the semantic clustering approach to identify whether the sentential *topics* entails the document level *topic* or not. The semantic clustering approach incorporates three types of similarity coefficients. The sentential emotion *topics* with higher valued coefficients are only considered as the potential document level *topics*. Additionally, the features like bigrams, Named Entities (NEs) and sentiments are incorporated to identify the document level topics and therefore improve the results of the semantic clustering. The improvement has also been found during the evaluation of holder-topic relations with respect to Ekman's six emotion classes. The evaluation of many-to-many relationships between ten *bloggers* and eight *topics* achieves the average precision, recall and F-Score of 65.02 %, 76.23 % and 70.18 % for ten bloggers and 71.02 %, 78.47 % and 74.55 % with respect to eight topics, respectively.

The remainder of the paper is organized as follows: in the next section, we review the related works. The baseline and supervised system frameworks are elaborated in Sect. 3 with experiments and error analysis. The document level topic identification is discussed in Sect. 4. Section 5 proposes a brief description of semantic clustering approach for measuring similarity between sentence and document level *topics*. Evaluation from the perspectives of the *holders* and *topics* are discussed in Sect. 6. Finally, Sect. 7 concludes the paper.

2 Related Work

Human-machine interface technology has been investigated for several decades and scientists have found that emotion technology can be an important component in artificial intelligence [16]. In recent times, research activities in the areas of emotion in natural language texts and other media are gaining ground under the umbrella of subjectivity analysis and affect computing. The reason may be the explosive growth of the social media content on the Web in the past few years.

In order to estimate affects in text, the model proposed in [17] processes symbolic cues and employs natural language processing techniques for word/phrase/sentence

level analysis by considering relations among words in a sentence. The current trend in the emotion analysis area is exploring machine learning techniques that consider the problem as text categorization or analogous to topic classification [18] and underscores the difference between machine learning methods and human-produced baseline models [19]. The affective text shared task on news headlines at *SemEval* 2007 for emotion level identification has drawn the focus to this field [20].

In case of emotion *holder*, the prior work has sometimes identified only a single opinion per sentence and sometimes several [21]. The identification of opinion *holder*s for Question Answering with supporting annotation task was attempted in [2]. Based on the traditional perspectives, another work discussed in [22] uses an emotion knowledge base for extracting *emotion holder*. Kim and Hovy [11] identified opinion *holder* with topic from media text using semantic role labeling. The model extracts opinion topics associated with specific argument position for subjective expressions signaled by verbs and adjectives. The syntactic models of identifying *emotion holder* with respect to emotional verbs were developed in [23]. Similarly, the verb based argument extraction and associated topic identification has been considered for developing our present system of identifying *emotion holders*.

In the related area of opinion topic extraction, different researchers have contributed their efforts. But, most of the works are based on lexicon look up approaches and applied on the domain of product reviews. The topic annotation task on the MPQA corpus is described in [24]. The authors mentioned that the opinion topics are not necessarily spatially coherent as there may be two opinions in the same sentence on different topics, as well as opinions that are on the same topic separated by opinions that do not share that topic. In contrast, the building of fine-grained topic knowledge based on rhetorical structure theory and segmentation of topics using different types of lexical, syntactic and overlapping features in our supervised framework substantially reduces the problem of emotion topic distinction.

Moreover, all the above cited works have been attempted for English. Recent study shows that non-native English speakers support the growing use of the Internet.[1] In addition to that, rapidly growing web users from multilingual communities have focused the attention to improve the multilingual search engines on the basis of sentiment or emotion. This raises the demand of emotion analysis for languages other than English. The Bengali is the sixth popular language in the World,[2] second in India and the national language in Bangladesh but it is less computerized compared to English. Works on emotion analysis in Bengali have started recently [25]. The comparative evaluation of the features on Bengali and English blog data can be found in [26]. To the best of our knowledge, at present, no such task on user-topic based emotion analysis has been conducted for Bengali or even for other Indian languages. Thus, we believe that this topic focused emotion analysis systems along with their associated holders would contribute in other social networking applications.

[1] http://www.internetworldstats.com/stats.htm.

[2] http://www.ethnologue.com/ethno_docs/distribution.asp?by=size.

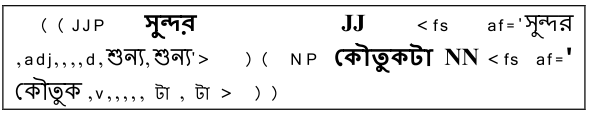


Fig. 1 Example of a preprocessed shallow parsed result

3 System Framework

3.1 Baseline System

The baseline system assumes that *emotional expression*, *holder* and *topic* appear as the neighboring chunks. The blog sentences are passed through an open source Bengali shallow parser[3] that gives different morphological information and clues (*root*, *lexical category of the root*, *gender*, *number*, *person*, *case*, *vibhakti*, *tam*, *suffixes* etc.) in identifying the lexical patterns of *emotional expressions* (e.g., the shallow parsed *emotional expression* is shown in Fig. 1). We search each component word of a chunk in the Bengali *WordNet Affect* [13]. If any word present in a chunk is an emotion word (e.g. কৌতুক *koutuk* 'comic'), all of the words present in that chunk are then treated as the candidate seeds for an *emotional expression*. Each of the *emotional expressions* is tagged with one of the Ekman's six emotions based on the type of the Bengali *WordNet Affect* lists in which its responsible component word appears.

The baseline system considers the phrasal pattern containing similarity clues to identify the emotion *holders*. The patterns are grouped according to the part-of-speech (POS) categories. It is observed that the hints of grouping the lexical patterns present mostly in the sentences of user comments. The Bengali blog structure[4] is well formed and each of the user comment sections starts with a corresponding username that is assumed as a default hint in capturing the *holder* information. But, a sentence may contain more than one emotion *holder* if we consider the nested source hypothesis [2]. In such cases, the POS tags of the shallow parsed sentences contain the similarity patterns at lexical level. In case of simple sentences, the similarity pattern consists of two phrasal constituents, the subject and verb. The portion of a sentence excluding the subject and verb contains only additional constituents that are grouped to form a common portion (*Common_Portion*). As the Bengali is a free phrase as well as word order language, the ordering between the subject, verb and the *Common_Portion* is not fixed. The words that are tagged as Named Entities (NEs), NNPC (Compound proper noun), NNP (Proper noun), NNC (Compound common noun), NN (Common noun) or PRP (Pronoun) and occur in the beginning of the sentences are considered as the emotion *holders*.

The general POS level pattern such as [<NNP/NNPC/NN/NNC/PRP> {<VBZ/VM> <*Common_Portion*>}] has been considered for capturing the clue of an emotion *holder*. The components of the *Common_Portion* are assembled after

[3] http://ltrc.iiit.ac.in/showfile.php?filename=downloads/shallow_parser.php.

[4] www.amarblog.com/.

the first occurring POS tags of types NNP or NNC or PRP in the POS tagged sentence. The process stops by reaching the verb POS, like VBZ or VM. The rest of any component words that are present in a sentence are appended to form the *Common_Portion*. It has been observed that the similarity patterns are difficult to obtain from complex or compound sentences using the baseline system. The system fails to identify the nested emotion *holder*s present in different clausal segments of a sentence but identifies one or more potential emotion *topics* from the shallow chunks by removing the *emotional expressions* and *holders*. The emotion *topic* is intended by its emotion *holder* and it depends on the context in which its associated *emotional expression* occurs [24]. Hence, the words of POS types such as NNP or NNC that are present in the neighboring shallow chunks of the *emotional expressions* and *holders* are tagged as the emotion *topics*.

3.2 SVM Based Supervised System

In order to improve the results of our baseline system, the Support Vector Machine (SVM) [14] based supervised framework has been adopted to extract the *emotional expressions*, *holders* and *topics*. Three separate modules have been developed for identifying these three components. Feature plays a crucial rule in the SVM framework. By manually reviewing the blog data and different language specific characteristics, the following word level as well as context level features have been selected heuristically. Training with 2225 sentences, the best features identified from 630 development sentences have been reused in test set. Some extra features that were adopted for handling the error cases have also been used in training and test phases.

A total of 3367 emotional sentences on eight different potential *topics* (Comics, Politics, Sports, Movies, Music, Buzz, Short Stories and Miscellaneous) with respect to 10 different blog users were considered for conducting the experiments. The sentences were collected from an annotated corpus [13]. Each sentence of the corpus is annotated with the emotional components such as *emotional expression* (word/phrase), associated *holder* and *topic*(s). Ekman's six emotion tags (*anger*, *disgust*, *fear*, *happy*, *sad* and *surprise*) are annotated at sentence level.

Lexical Features Several lexical features like ***Parts-of-Speech*** (**POS**) [*noun*, *adjective*, *verb* and *adverb* words], ***Negations*** (**NEG**) [annotated negative words (*noy* 'not', *na* 'neither' etc.)], ***Conjuncts*** (**CONJ**) [annotated *conjuncts* (e.g. *ebong* 'and', *athoba* 'or', *kintu* 'but' etc.)], ***Punctuation Symbols*** (**Sym**) [single or multiple numbers of symbols like (,), (!), (?)], ***Named Entity*** (**NE**) [*person* or *organization* type entities identified using Named Entity Recognizer [27] have been used as holder], ***Emoticons*** (**emot_icon**) [(☺, ☹, 😐) and their consecutive occurrence generally contribute real sentiment to the words that precede or follow it].

Syntactic Features Specially, the subcategorization frames that are identified from the shallow parsed Bengali sentences. A rule based *phrasal-head* extraction

module is used to identify the phrase level argument structures with respect to the verbs. The extracted *head part* of every phrase from a parsed sentence is considered as a component of its sentential argument structure. The acquired argument structure for a Bengali emotional sentence is matched with any of the available extracted frames of English VerbNet for the equivalent English verbs corresponding to that Bengali verb. If any match is found, the *thematic roles* for holder (e.g. *Experiencer*, *Agent*, *Actor*, *Beneficiary* etc.) and *topic* (e.g. *Topic*, *Theme*, *Event* etc.) associated with the English frame syntax is mapped to the appropriate slot of the acquired Bengali argument structure. Equivalent English verbs are identified using Bengali to English bilingual dictionary.[5] The hypothesis that was considered in [28] has been used in our present task. Instead of the ordering dissimilarity, the phrase level similarity between English and Bengali languages helps in identifying the subcategorization frames. The case markers in Bengali are required to identify the emotion *holder*s as the case markers give the useful hints to capture the *selectional restrictions* that play a key role in distinguishing the emotion *holder*s.

রাশেদ অনুভব করেছিল যে রামের সুখ অন্তহীন
(*Rashed*) (*anubhob*) (*korechilo*) (*je*) (*Ramer*) (*sukh*) (*antohin*)
Rashed felt that Ram's pleasure is endless.
Holder : < EH_রাশেদ, EH_*রাম*, ET_সুখ >
Acquired Argument Structure: [**NNP** VM DET-$_{je}$ S]
Equivalent English Verb: *feel*
Extracted VerbNet Frame Syntax: [< **NP** value = "***Experiencer***" >< /VERB >
< S-$_{that}$ (Sentential$_{-that}$ Complement) >]

Semantic Features One of the Semantic features is ***Emotion/Affect Word*** (**EW**) present in the Bengali *WordNet Affect* lists [13]. The Bengali *SentiWordNet* is being developed by replacing each word entry in the synonymous set of the English *SentiWordNet* [29] by its possible Bengali synsets using the English to Bengali bilingual dictionary. The shallow chunks containing JJ (adjective) and RB (adverb) tagged word elements are considered as intensifiers. If the intensifier is found in the Bengali *SentiWordNet*, the positive and negative scores of the intensifier are retrieved. The intensifier is classified into the list of positive (pos) (**INTF*pos***) or negative (neg) (**INTF*neg***) for which the average retrieved score is higher. One of the important semantic features is ***Multiword Expressions*** [*Reduplication* (*sanda sanda* [doubt with fear]) and *Idioms* (*taser ghar* [weakly built], *grrihadaho* [family disturbance])].

Rhetoric Features Present task acquires the basic rhetorical components such as *locus*, *nucleus* and *satellite* [30] from a sentence as these rhetoric clues help in identifying the individual *topic* spans. The part of the text span containing annotated *emotional expression* is considered as *locus*. Primarily, the separation of *nucleus* from *satellite* is done based on the punctuation markers (,) (!) (?). Frequently used *discourse markers* (*jehetu* 'as', *jemon* 'e.g.', *karon* 'because', *mane* 'means') and

[5] http://home.uchicago.edu/~cbs2/banglainstruction.html.

Table 1 *F-Scores* of *emotional expression*, *holder* and *topic* for different features on development set

Features	Emotional expression	Holder	Topic
Baseline System	22.80 %	21.89 %	18.78 %
Supervised System			
Lexical Features	41.88 %	30.78 %	21.56 %
Syntactic Features	44.67 %	20.09 %	32.99 %
Semantic Features	46.95 %	12.04 %	16.07 %
Rhetoric Features	33.05 %	21.80 %	31.02 %
Lexical + Semantic Features	63.87 %	25.67 %	45.21 %
Syntactic + Rhetoric Features	57.69 %	51.78 %	51.90 %
Lexical + Syntactic + Semantic Features	67.77 %	59.43 %	50.66 %
Lexical + Semantic + Rhetoric Features	63.86 %	57.27 %	52.89 %
Lexical + Syntactic + Semantic + Rhetoric Features	**72.54 %**	**61.02 %**	**57.09 %**

causal verbs (*ghotay* 'caused') also act as the useful clues if they are explicitly specified in text. Stoyanov and Cardie mentioned in [29] that *topic* depends on the context in which its associated *emotional expression* occurs. Based on this hypothesis, we assumed that if any word of an *emotional expression* co-occurs with any word element of the *nucleus* or *satellite* in the same shallow chunk, the feature is considered as *common rhetoric similarity*. Otherwise, the feature is considered as *distinctive rhetoric similarity*. This two features aim to separate emotion *topics* from non-emotion *topics* as well as the individual *topic* from overlapped spans.

3.3 Experiments

It is found that the supervised system not only outperformed the baseline system but the combination of multiple features in comparison with a single feature in the supervised system also shows the reasonable performance enhancement of the classification system. The results have been shown in Table 1. Thus, the impact of different features and their combinations were measured on the development set and the results are shown in Table 1. We have added each feature into the active feature list one at a time if the inclusion of the feature in the existing feature set improves the *F-Score* of the system. The final active feature set has been applied on the test set. Different unigram and bi-gram context features (word, POS tag, Intensifier, negation) and their combinations were generated from the training corpus. The emotion word, POS, intensifier features have played an important role in extracting *emotional expressions*. In SVM-based training phase, the current token word with three previous and three next words and their corresponding POS along with negation or intensifier were selected as context feature for that word. Specially, the syntactic and rhetoric features help in identifying the *holder* and *topic* whereas semantic feature

contributes for identifying the *emotional expressions*. Overall, the average F-scores of the supervised system are 72.54 %, 61.02 % and 57.09 % for *emotional expression*, *holder* and *topic* respectively on 512 test sentences. It has to be mentioned that the results produced by the supervised system are satisfactory in comparison with the agreement values of the subjective annotation for the above three emotion components. The difficulty faced in case of satisfying few implicitly specified components.

4 Document Level Topic Identification

The topic is a real world object, event or an abstract entity that is the primary subject of the opinion/emotion as intended by the holder [24]. There has been extensive research on automatic text analysis for affect [31, 32]. But, sometimes, the document level emotion classification fails to detect emotion about individual aspects of the topic. For example, though one could be generally happy about his car, he might be dissatisfied by the engine noise. To the manufacturers, these individual weaknesses and strengths are equally important to know, or even more valuable than the overall satisfaction level of customers. On the other hand, the topic identification is also essential in connection within categorizing search applications, where several sets of documents are delivered and an expressive description for each category must be constructed on the fly.

In the related previous tasks, several statistical methods have been used for topic identification, such as topic datagram, TF-IDF, cache or weighted unigrams etc. In contrast, we follow a combined approach consisting of bigram count, Named Entity and sentiment words to identify the document level topics. Though these features were already considered during sentence level emotion classification, their utilization was highly noticed during document level emotion tagging. The diagram of the combined module is shown in Fig. 2.

We have removed the stop words from the corpus using stop word list that contains 320 stop words. Stop words are common "*function words*" such as *ei* '*the*', *ki* '*what*' etc. Some special symbols were also removed from the corpus like *braces*.

Separately, a knowledge base (as shown in Table 2) for the emoticons was also prepared by experts after minutely analyzing the Bengali blog data. Each image link of the emoticon in the raw corpus was mapped into its corresponding textual entity in the tagged corpus according to their proper emotion types using this knowledge base. The knowledge base has also been used during the emotion tagging of the sentences by considering each of the emoticons as a separate word.

4.1 Bigram Count

In the fields of *computational linguistics* and *probability*, an n-gram is a contiguous sequence of n items from a given sequence of text. An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a

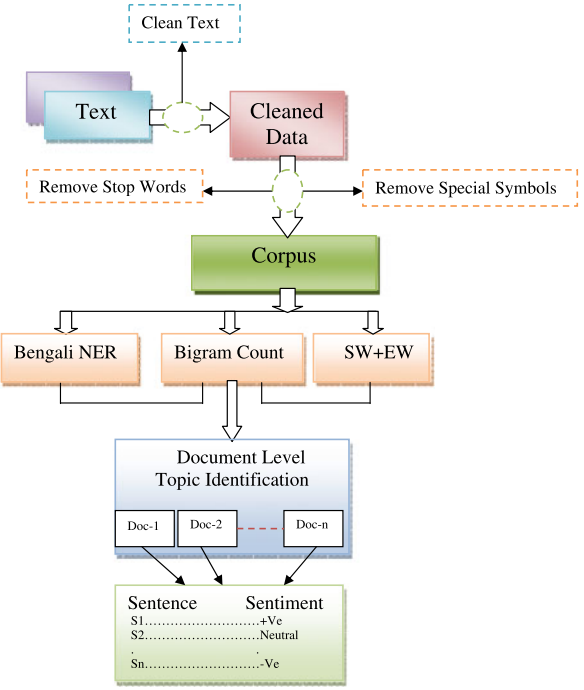


Fig. 2 System diagram for identifying document level topics

Table 2 Knowledge base for emoticons

Emoticon	Tags	
☺, :-)	<emo_icon_happy>	*happy*
☹, :-S	<emo_icon_sad>	*sad*
:-@, :-a	<emo_icon_ang>	*anger*
:-$, :-D	<emo_icon_dis>	*disgust*
:'(, :-F	<emo_icon_fear>	*fear*
:-O, :-P	<emo_icon_sur>	*surprise*
😐, :-|	<emo_icon_ntrl>	*neutral*

"trigram" and so on. Primarily, it has been observed that the unigrams fail to produce complete topic names and trigram adds extra noise in identified topic names whereas the bigram count produces satisfactory results in identifying complete topic names with less amount of noise.

4.2 Named Entity

We have used a Bengali Named Entity Recognizer (NER) [27] for identifying the Named Entities related to topic names from the texts of a blog document. The

NER labels sequences of words in a text such as *person*, *location* and *organization/company* names. We have retrieved all the Named Entities from the blog documents and employed them in the combined module.

4.3 Sentiment and Emotion Words

We have used two resources for identifying sentiment words within a blog document. The emotion words (EW) of the Bengali *WordNet Affect* lists [13] have been used for identifying the sentiment words of a document. The Bengali *SentiWordNet* is being developed by replacing each word entry in the synonymous set of the English *SentiWordNet* [33] by its possible Bengali synsets using the English to Bengali bilingual dictionary that was developed as part of the EILMT project.[6] We assumed that the topic words present in the context where its associated sentiment or *emotional expression* occurs [24]. Thus, we have identified the sentiment sentences and employed them in the combined module.

4.4 Combined Module

We have measured the performance of each module separately and in combination with other module(s). It has been observed that bigram count along with sentiment words performs better in comparison with the combined module consisting of bigrams and Named Entity. The performance of each module and the combined performances of the modules have been evaluated manually in terms of accuracy. The total number of blog documents for testing was 82. Overall, the combination of all the three modules achieves an accuracy of 67.34 % for document level topic identification. The results are shown in Table 3.

Table 3 Accuracies (in %) of the individual module and combined modules on the test set

Module combinations	Accuracy
Bigram Count	31.14 %
NER	24.22 %
Sentiment Words (SW) + Emotion Words (EW)	26.15 %
Bigram + NER	43.05 %
Bigram + SW + EW	53.90 %
NER + SW + EW	47.69 %
Bigram + NER + SW + EW	67.34 %

[6]English to Indian Languages Machine Translation (EILMT) is a TDIL project undertaken by the consortium of different premier institutes and sponsored by MCIT, Govt. of India.

5 Semantic Clustering for Identifying Topic Similarity Between Document and Sentence Level Topics

Now, it is clear the structure of the blog documents. Each of the blog documents contains a topic (document level) followed by different sections for the bloggers' comments. It is observed that the *topics* discussed by the bloggers in their comment sections especially at sentence level are not similar with the *topic* that is described at the document level. Sometimes, the *topics* in the sentential comments of the bloggers are completely irrelevant with the *topic* of its corresponding document. Hence, a semantic clustering approach has been adopted for clustering the semantically related topic words present in a document by incorporating the features of bigram, NE and sentiment into account.

We have seen that these features are the important clues for identifying document level topics. Thus, we have applied the document level topic identification module for identifying the potential candidate word(s) or token from the topic sections of the blog documents. The tokens which are tagged by any of the three modules, bigram, NE or sentiment are termed as the candidate tokens and have been considered for semantic clustering. Identifying semantically related words for a particular candidate token is carried out by looking the surrounding tokens and finding the synonymous words within a fixed context window. Higher value of the similarity coefficient between two synsets of the target words indicates more affinity of the words to each other. For individual word, semantically related words of the documents are extracted by using a monolingual dictionary.[7] Count of elements in an intersection of two synsets indicates the commonality of the two sets and its absolute value stands for the commonality measure. Considering the common elements as the dimensions of the vector space, similarity based techniques are applied to measure the semantic affection of two target words.

In the first phase, we have generated the synsets for all noun words present in the corpus using the synset based dictionary whereas in the second phase, the task is to identify the semantic distance between two nouns. The format of the dictionary is as follows,

$$\begin{aligned} W^1 &= n_1^1, n_2^1, n_3^1, \ldots = \left\{n_i^1\right\} \\ &\vdots \\ W^m &= n_1^m, n_2^m, n_3^m, \ldots = \left\{n_p^m\right\} \end{aligned}$$

where, $W^1, W^2, \ldots, W^m$ are the dictionary word entries and n_i^m (for all i) are the elements of the synsets of W^m. Now, each noun entry identified by the shallow parser in the document is searched in the dictionary. If N, a noun in the corpus is present in the synsets of W^1, W^3 and W^5, the retrieved synset for N is as follows,

$$\text{SynSet}(N) = \left\{W^1, W^3, W^5\right\}. \tag{1}$$

[7] http://dsal.uchicago.edu/dictionaries/biswas-bangala/.

Table 4 Cut off and *F-Scores* (in %) for *topic* identification using three different measures

Cut-off	F-Score (in %)		
	Cosine-similarity	Euclidean distance	WordNet similarity
0.6	67.68	66.14	67.58
0.5	67.74	65.12	68.06
0.4	64.08	63.71	61.63

To identify the semantic similarity between two nouns, we have applied simple intersection rule. The number of common elements between the synsets of the two noun words denotes the similarity between them. If N_i and N_j are the two noun words in the document and W_i and W_j are their corresponding synsets, the similarity of the two words can be defined as:

$$\text{Similarity}(N_i, N_j) = |W_i \cap W_j|. \tag{2}$$

We have clustered all the nouns words present in the document for a particular noun word and give a score using three similarity techniques. The results of the similarity techniques based on cut-off values are shown in Table 4.

ALGORITHM: TOPIC CHECKING

INPUT: Two Word Tokens <M1 M2>

OUTPUT: Return true if *Topic*, or return false.

1. Extract semantic clusters of M1 and M2.
2. Intersection of the clusters of both M1 and M2 (Fig. 3 shows the common synset entries of M1 and M2 using rectangle).
3. Measuring the semantic similarity between M1 and M2:
   3.1. In an n-dimensional vector space (here n = 2), the common entries act as the axes. Put M1 and M2 as two vectors and associated weights as their co-ordinates.
   3.2. Calculate cosine-similarity measurement and Euclidean distance (Fig. 3).
4. Final decision taken individually for two different measurements
   4.1. If cosine-similarity > m, return false; Else return true;
   4.2. If Euclidean distance > n, return false; Else return true;

(Where m and n are the pre-defined cut-off values).

In addition to the cosine-similarity and the similarity based on Euclidean distance, we have employed the English WordNet[8] to measure the semantic similarity between two Bengali words translated into English. WordNet::Similarity is an open-source package for calculating the lexical similarity between word (or sense) pairs based on variety of similarity measures. We have translated the root of the two components of a Bengali candidate into their English equivalents.

[8] http://www.d.umn.edu/tpederse/similarity.html.

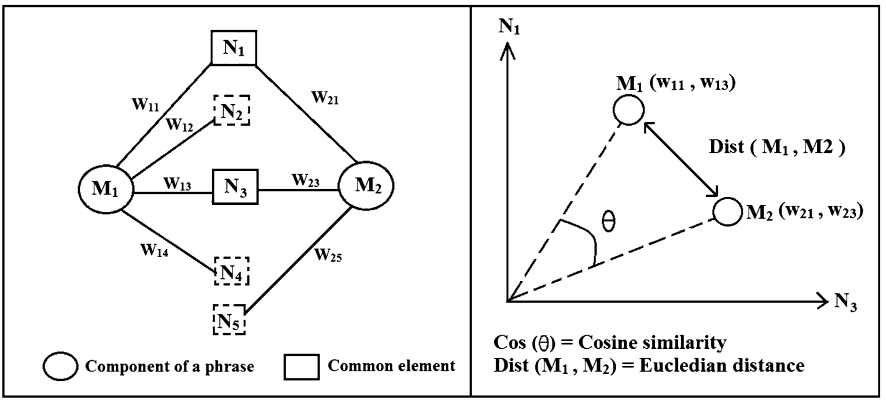


Fig. 3 Intersection of the clusters of the constituents and similarity between two constituents

In addition to the above issues, another contribution to the semantic clustering was the application of the candidate tokens. We have identified the candidate tokens from the topic section of the blog documents by applying the combined module of bigram, NE and sentiment. It has been observed that the results of the semantic clustering approach have been improved satisfactorily by choosing only the candidate tokens rather than employing all of the words from the document level topic section of the blogs.

6 Evaluation of Holder-Topic

The sentiment *topics*/targets should not be done alone without considering sentiment holders. A single *topic* is referred by several users as well as multiple *topics* are referred by a single user. The user-topic relations aim to generate many to many correspondences. Hence, we have adopted two way evaluation process, one is from the perspective of the users and another is from the perspectives of topics.

The *emotional expression*, *holder* and *topic* are identified and tagged accordingly on 512 test sentences that include the equal distribution of ten blog users' sentential comments on eight different potential *topics*. The many-to-many relationships between ten *bloggers* and eight *topics* have been analyzed with respect to Ekman's (1993) six emotion classes. The Ekman's six different emotions are plotted for 8 different topics referred by each of the 10 bloggers. The *topic* based emotions of the bloggers as shown in Fig. 5 signifies that the system achieves high *F-Score* with respect to each of the topics. It was observed that the emotional views of the bloggers and its dependence on the associated *topics* achieves significantly better results in identifying user and *topic* specific emotions in a particular time period. The charts in Figs. 4 and 5 show the *F-Score* values (in %) for Ekman's six emotions from the perspectives of users and topics. The system achieves the average *precision*, *recall* and *F-Score* of 65.02 %, 76.23 % and 70.18 % for ten bloggers and

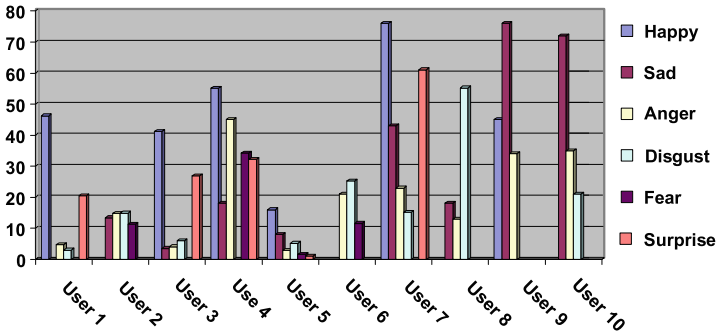
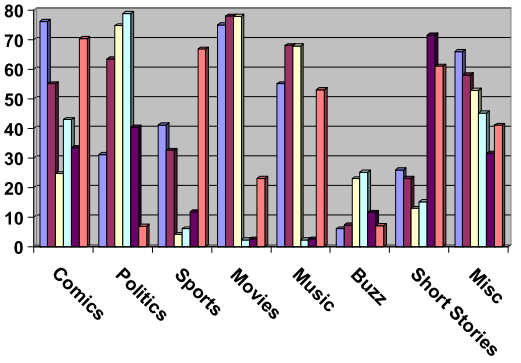


Fig. 4 F-Scores (in %) for Ekman's six emotions from the perspectives of users

Fig. 5 F-Scores (in %) for Ekman's six emotions from the perspectives of *topics*



71.02 %, 78.47 % and 74.55 % with respect to eight topics, respectively on 512 test sentences.

7 Conclusion

The present task identifies the emotions of the bloggers on different topics provided in the Bengali blog documents using two prong approaches, a rule based system followed by a supervised one. The contribution of the present task is to classify the blog documents in a meaningful way by employing the features of bigram, NE and sentiment and associate the document level topics with the sentential topics using semantic clustering. The present research will help in categorizing of the blog documents based on topics from the perspectives of relevancy as well as consistency on the overall theme of the documents. The handling of metaphors and their impact in detecting sentence level emotion is not considered. Future analysis concerning the time based emotional change can be used for *topic* model representation. The need of co reference entails that the presence of indirect affective clues can also be traced with the due help of *holder* and *topic*.

Acknowledgements The work reported in this paper was supported by a grant from the India-Japan Cooperative Programme (DSTJST) 2009 Research project entitled "Sentiment Analysis where AI meets Psychology" funded by Department of Science and Technology (DST), Government of India.

References

1. Sood S, Vasserman L (2009) ESSE: exploring mood on the web. In: 3rd international AAAI conference on weblogs and social media data challenge workshop
2. Wiebe J, Wilson T, Cardie C (2005) Annotating expressions of opinions and emotions in language. Lang Resour Eval 39(2–3):165–210
3. Lin KH-Y, Yang C, Chen H-H (2007) What emotions news articles trigger in their readers? In: SIGIR, pp 733–734
4. Yang C, Lin KHY, Chen HH (2007) Emotion classification using web blog corpora. In: IEEE, WIC, ACM international conference on web intelligence, pp 275–278
5. Cardie C, Wiebe J, Wilson and T, Litman JD (2003) Combining low-level and summary representations of opinions for multi-perspective question answering. In: New directions in question answering, pp 20–27
6. Pang B, Lee L (2008) Opinion mining and sentiment analysis. Found Trends Inf Retr 2(1–2):1–135
7. Yang C, Lin KHY, Chen HH (2009) Writer meets reader: emotion analysis of social media from both the writer's and reader's perspectives. In: IEEE/WIC/ACM international joint conference on web intelligence and intelligent agent technology, pp 287–290
8. Fukuhara T, Nakagawa H, Nishida T (2007) Understanding sentiment of people from news articles: temporal sentiment analysis of social events. In: ICWSM
9. Das D, Kolya A, Ekbal A, Bandyopadhyay S (2011) Gelbukh A (ed) Temporal analysis of sentiment events – a visual realization and tracking, Tokyo. LNCS, vol 6608, pp 417–428
10. Seki Y (2007) Opinion holder extraction from author and authority viewpoints. In: SIGIR'07. ACM, New York, 978-1-59593-597-7/07/0007
11. Kim SM, Hovy E (2006) Extracting opinions, opinion holders, and topics expressed in online news media text. In: Workshop on sentiment and subjectivity, ACL
12. Das D, Bandyopadhyay S (2010) Labeling emotion in Bengali blog corpus – a fine grained tagging at sentence level. In: 8th workshop on Asian language resources (ALR8), COLING-2010, pp 47–55
13. Das D, Bandyopadhyay S (2010) Developing Bengali WordNet affect for analyzing emotion. In: ICCPOL-2010, California, pp 35–40
14. Joachims T (1998) Text categorization with support machines: learning with many relevant features. In: European conference on machine learning, pp 137–142
15. Stein B Meyer zu Eissen S (2004) Topic identification: framework and application. In: Tochtermann M (ed) Proceedings of the I-KNOW'04, Graz 4th international conference on knowledge management, J Univers Comput Sci, pp 353–360. ISSN 0948-6968
16. Salovey P, Mayer J (1990) Emotional intelligence. Imagin Cogn Pers 9(3):185–211
17. Neviarouskaya A, Prendinger H, Ishizuka M (2007) Narrowing the social gap among people involved in global dialog: automatic emotion detection in blog posts. In: ICWSM
18. Sebastiani F (2002) Machine learning in automated text categorization. ACM Comput Surv 34(1)
19. Alm CO, Roth D, Sproat R (2005) Emotions from text: machine learning for text-based emotion prediction. In: HLT-EMNLP, pp 579–586
20. Strapparava C, Mihalcea R (2007) SemEval-2007 task 14: affective text. ACL
21. Bethard S, Yu H, Thornton A, Hatzivassiloglou V, Jurafsky D (2004) Automatic extraction of opinion propositions and their holders. In: AAAI spring symposium on exploring attitude and affect in text: theories and applications

22. Hu J, Guan C, Wang M, Lin F (2006) Model of emotional holder. In: Shi Z-Z, Sadananda R (eds) PRIMA 2006. LNCS (LNAI), vol 4088, pp 534–539
23. Das D, Bandyopadhyay S (2010) Emotion holder for emotional verbs – the role of subject and syntax. In: Gelbukh A (ed) CICLing-2010. LNCS, vol 6008, pp 385–393
24. Stoyanov V, Cardie C (2008) Annotating topics of opinions. In: LREC
25. Das D, Bandyopadhyay S (2009) Word to sentence level emotion tagging for Bengali blogs. In: ACL-IJCNLP, 2009, pp 149–152
26. Das D, Bandyopadhyay S (2010) Sentence level emotion tagging on blog and news corpora. J Intell Syst (JIS) 19(2):125–134
27. Ekbal A, Bandyopadhyay S (2010) Named entity recognition using appropriate unlabeled data, post-processing and voting. Informatica (Slovenia) 34(1):55–76
28. Banerjee S, Das D, Bandyopadhyay S (2010) Classification of verbs – towards developing a Bengali verb subcategorization lexicon. In: GWC-2010, pp 76–83
29. Andrea E, Sebastiani F (2006) SENTIWORDNET: a publicly available lexical resource for opinion mining. In: LREC
30. Mann WC, Thompson SA (1998) Rhetorical structure theory: toward a functional theory of text organization. Text 8:243–281
31. Subasic P, Huettner A (2001) Affect analysis of text using fuzzy semantic typing. IEEE Trans Fuzzy Syst, Special Issue Aug
32. Whissell C (1989) The dictionary of affect in language. In: Emotion: theory, research, and experience, pp 113–131
33. Baccianella S, Esuli A, Sebastiani F (2010) SentiWordNet 3.0: an enhanced lexical re-source for sentiment analysis and opinion mining. In: LRE, pp 2200–2204

Predicting Number of Zombies in a DDoS Attacks Using Isotonic Regression

B.B. Gupta and Nadeem Jamali

Abstract Anomaly based DDoS detection systems construct profile of the traffic normally seen in the network, and identify anomalies whenever traffic deviate from normal profile beyond a threshold. This deviation in traffic beyond threshold is used in the past for DDoS detection but not for finding number of zombies. This chapter presents an approach that utilizes this deviation in traffic to predict number of zombies using isotonic regression model. A relationship is established between number of zombies and observed deviation in sample entropy. Internet type topologies used for simulation are generated using Transit-Stub model of GT-ITM topology generator. NS-2 network simulator on Linux platform is used as simulation test bed for launching DDoS attacks with varied number of zombies. Various statistical performance measures are used to measure the performance of the regression model. The simulation results are promising as we are able to predict number of zombies efficiently with very less error rate using isotonic regression model.

Keywords DDoS attack · Intrusion detection · Isotonic regression · Zombies · Entropy

1 Introduction

DDoS attacks are an impending threat to Internet related applications. It poses a grave danger to users, organizations and infrastructures of the Internet. A DDoS attacker attempts to disrupt a target, in most cases a web server, by flooding it with illegitimate packets, usurping its bandwidth and overtaxing it to prevent legitimate inquiries from getting through [1, 2]. In anomaly based DDoS detection mechanisms, the profile of the traffic normally seen in the network is constructed

B.B. Gupta (✉)
Indian Institute of Technology Roorkee, Roorkee, India
e-mail: gupta.brij@gmail.com

N. Jamali
University of Saskatchewan, Saskatoon, Canada
e-mail: jamali@agents.usask.ca

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_8,

and anomalies are identified whenever traffic deviates from normal profile beyond a threshold [3]. Proposed approach utilizes this deviation in traffic beyond threshold to predict number of zombies using isotonic regression [4, 5]. A real time estimation of the number of zombies in DDoS scenario is helpful to suppress the effect of attack by choosing predicted number of most suspicious attack sources for either filtering or rate limiting. Moore et al. [6] have already made a similar kind of attempt, in which they have used backscatter analysis to estimate number of spoofed addresses involved in DDoS attack. This is an offline analysis based on unsolicited responses.

Our objective is to find the relationship between number of zombies involved in a flooding DDoS attack and deviation in sample entropy. In order to predict number of zombies, isotonic regression model is used. To measure the performance of the proposed approach, we have calculated various statistical performance measures i.e. R^2, CC, SSE, MSE, RMSE, NMSE, η, MAE and residual error. Internet type topologies used for simulation are generated using Transit-Stub model of GT-ITM topology generator [7]. NS-2 network simulator [8] on Linux platform is used as simulation test bed for launching DDoS attacks with varied number of zombies. In our simulation experiments, attack traffic rate is fixed to 25 Mbps in total; therefore, mean attack rate per zombie is varied from 0.25 Mbps to 2.5 Mbps and total number of zombie machines range between 10 and 100 to generate attack traffic.

The remainder of the chapter is organized as follows. Section 2 presents related work. Section 3 contains overview of isotonic regression model. Section 4 presents various statistical performance measures. Intended analytical model and detection scheme are described in Sect. 5. Section 6 describes experimental setup in details. Model development and performance analysis is presented in Sect. 7. Section 8 contains simulation results and discussion. Finally, Sect. 9 concludes the chapter.

2 Related Work

This section charts out the overview on a plethora of existing DDoS defense schemes proposed in the literature. Exiting DDoS defense schemes are classified into four broad categories: Prevention, Detection, Response, and Tolerance and Mitigation. Attack prevention methods try to stop all well known signature based and broadcast based DDoS attacks from being launched in the first place or edge routers, keeps all the machines over Internet up to date with patches and fix security holes. The approaches to stop IP spoofing [9], filtering malicious IP addresses based on experience [10], Remove unused services [11] and repairing security holes by patches [12] fall under this category. Attack prevention schemes are not enough to stop DDoS attacks because these are always vulnerable to novel and mixed attack types for which signatures and patches do not exist in the databases. Attack detection aims to detect an ongoing attack and to discriminate malicious traffic from legitimate traffic. Detection can be performed using database of known signatures, by recognizing anomalies in system behaviors or using third party. Signature based

approach employs a priori knowledge of attack signatures. The signatures are manually constructed by security experts analyzing previous attacks and used to match with incoming traffic to detect intrusions. SNORT [13] and Bro [14] are the two widely used signature based detection approaches. Signature based techniques are only effective in detecting traffic of known DDoS attacks whereas new attacks or even slight variations of old attacks go unnoticed. Anomaly detection [15–21] relies on detecting behaviors that are abnormal with respect to some normal standard.

Gil and Poletto [15] proposed a scheme called MULTOPS to detect denial of service attacks by monitoring the packet rate in both the up and down links. MULTOPS assumes that packet rates between two hosts are proportional during normal operation. A significant disproportion between the packet rate going to and from a host or subnet is a strong indication of a DoS attack. Blazek et al. [16] proposed batch detection to detect DoS attacks by monitoring statistical changes. Cheng et al. [17] proposed to use spectral analysis to identify DoS attack flows. Lee and Stolfo [18] used data mining techniques to discover patterns of system features that describe program and user behavior and implement a classifier that can recognize anomalies and intrusions. A mechanism called congestion triggered packet sampling and filtering is proposed by Huang et al. [19]. According to this approach, a subset of dropped packets due to congestion is selected for statistical analysis. If anomaly is indicated by the statistical results, a signal is sent to the router to filter the malicious packets. Mirkovic et al. [20] proposed D-WARD defense system that does DDoS attack detection at source, based on the idea that DDoS attacks should be stopped as close to the source as possible. Bencsath et al. [21] have given a traffic level measurement based approach, in which incoming traffic is monitored continuously and dangerous traffic intensity rises are detected. Chen et al. [22] used distributed change-point detection (DCD) architecture using change aggregation trees (CAT) to detect DDoS attack over multiple network domains. Feinstein et al. [23] focus their detection efforts on activity level and source address distribution using entropy. Anomaly based techniques can detect novel attacks; however, it may result in higher false alarms. Mechanisms that deploy third-party detection do not handle the detection process themselves, but rely on an external third-party that signals the occurrence of the attack. Examples of mechanisms that use third-party detection are easily found among traceback mechanisms [24, 25].

The goal of the attack response is to relieve the impact of the attack on the victim while imposing minimal collateral damage to legitimate clients. The approaches to identify attack source/path or traceback [24, 25], filtering malicious traffic [26], and rate throttling malicious traffic [20, 27] fall under this category. Attack tolerance and mitigation focuses on minimizing the attack impact and tries to provide optimal level of service as per quality of its service requirement to legitimate users while service provider is under attack. The tolerance and mitigation solution includes router's queue management [28, 29], router's traffic scheduling [30], and target roaming [31].

In [32], authors have used linear regression and correlation analysis to predict number of zombies. But due to the nonlinear nature of DDoS attack traffic, this method is unable to predict the number of zombies accurately.

3 Isotonic Regression Model

Regression analysis [33, 34] is a statistical tool for the investigation of relationships between variables. Usually, the investigator seeks to ascertain the causal effect of one variable upon another. More specifically, regression analysis helps us to understand how the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held constant. Variables which are used to 'explain', other variables are called explanatory variables. Variable which is explained is called response variable. A response variable is also called a dependent variable, and an explanatory variable is sometime called an independent variable, or a predictor, or repressor. When there is only one explanatory variable the regression model is called a simple regression, whereas if there are more than one explanatory variable the regression model is called multiple regression.

Isotonic regression [4, 5] is also sometimes referred to as monotonic regression. Isotonic regression is used when the direction of the trend is strictly increasing, while monotonic could imply a trend that is either strictly increasing or strictly decreasing.

Isotonic regression (IR) involves finding a weighted least-squares fit $x \in \Re^n$ to a vector $a \in \Re^n$ with weights vector $w \in \Re^n$ subject to a set of monotonicity constraints giving a simple or partial order over the variables. The monotonicity constraints define a directed acyclic graph $G = (N, E)$ over the nodes $N = 1, 2, \ldots, n$ corresponding to the variables $x = x_1, x_2, \ldots, x_n$. Thus, the isotonic regression problem where a simple order is defined corresponds to the following quadratic program (QP):

$$\min \sum_{i=1}^{n} w_i (x_i - a_i)^2 \quad \text{subject to} \quad x_i \geq x_j \quad \forall (i, j) \in E \tag{1}$$

Isotonic regression has applications in statistical inference, for example, computing the cost at the minimum of the above goal function, gives the stress of the fit of an isotonic curve to mean experimental results when an order is expected. Isotonic Regression under the L_p for $p > 0$ is defined as follows:

$$\min \sum_{i=1}^{n} w_i |x_i - a_i|^p \quad \text{subject to} \quad x_i \geq x_j \quad \forall (i, j) \in E \tag{2}$$

Input and Output To predict number of zombies, we established relationship between number of zombies Y (output) and observed deviation in entropy X (input). For different given (known) zombies, deviation in sample entropy X is calculated as $(H_c – H_n)$, where H_c and H_n are entropy values at the time of attack detection and for normal profile, respectively. Regression equations are then determined by the process of curve fitting. These equations are used for predicting number of zombies.

4 Statistical Performance Measures

Various statistical performance measures, such as Coefficient of Determination (R^2), Coefficient of Correlation (CC), Standard Error of Estimate (SSE), Mean Square Error (MSE), Root Mean Square Error (RMSE), Normalized Mean square Error (NMSE), Nash–Sutcliffe Efficiency Index (η) and Mean Absolute Error (MAE) are used to evaluate the performance of the proposed regression model. These measures are defined below. In the definitions, N represents the number of feature vectors prepared, and denote the actual and the predicted values of dependent variable, respectively, and are the mean and the standard deviation of the actual dependent variable, respectively.

(i) *Coefficient of Determination* (R^2): Coefficient of determination (R^2) is a descriptive measure of the strength of the regression relationship, a measure how well the regression line fit to the data. R^2 is the proportion of variance in dependent variable which can be predicted from independent variable. The coefficient of determination (R^2) can be defined as:

$$R^2 = \frac{(\sum_{i=1}^{N}(Y_o - \overline{Y}_o)(Y_c - \overline{Y}_c))^2}{[\sum_{i=1}^{N}(Y_o - \overline{Y}_o)^2 \cdot \sum_{i=1}^{N}(Y_c - \overline{Y}_c)^2]} \tag{3}$$

(ii) *Coefficient of Correlation* (*CC*): The Coefficient of Correlation (CC) can be defined as:

$$\mathrm{CC} = \frac{\sum_{i=1}^{N}(Y_o - \overline{Y}_o)(Y_c - \overline{Y}_c)}{[\sum_{i=1}^{N}(Y_o - \overline{Y}_o)^2 \cdot \sum_{i=1}^{N}(Y_c - \overline{Y}_c)^2]^{1/2}} \tag{4}$$

(iii) *Sum of Squared Errors* (*SSE*): The Sum of Squared Errors (SSE) can be defined as:

$$\mathrm{SSE} = \sum_{i=1}^{N}(Y_o - Y_c)^2 \tag{5}$$

(iv) *Mean Square Error* (*MSE*): The Mean Square Error (MSE) between observed and computed outputs can be defined as:

$$\mathrm{MSE} = \frac{\sum_{i=1}^{N}(Y_c - Y_o)^2}{N} \tag{6}$$

(v) *Root Mean Square Error* (*RMSE*): The Root Mean Square Error (RMSE) between observed and computed outputs can be defined as:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(Y_c - Y_o)^2}{N}} \tag{7}$$

(vi) *Normalized Mean Square Error* (*NMSE*): The Normalized Mean Square Error (NMSE) between observed and computed outputs can be defined as:

$$\mathrm{NMSE} = \frac{\frac{1}{N}\sum_{i=1}^{N}(Y_c - Y_o)^2}{\sigma_{obs}^2} \tag{8}$$

(vii) *Nash–Sutcliffe efficiency index* (η): The Nash–Sutcliffe efficiency index (η) can be defined as:

$$\eta = 1 - \frac{\sum_{i=1}^{N}(Y_c - Y_o)^2}{\sum_{i=1}^{N}(Y_o - \overline{Y}_o)^2} \tag{9}$$

(viii) *Mean absolute error* (*MAE*): Mean absolute error (MAE) can be defined as follows:

$$\text{MAE} = 1 - \frac{\sum_{i=1}^{N}|Y_c - Y_o|}{\sum_{i=1}^{N}|Y_o - \overline{Y}_o|} \tag{10}$$

5 Detection of Attacks

5.1 Analytical Model

This section describes an analytical model which is constructed to detect a wide range of flooding attacks. Our approach detects flooding DDoS attacks by the constant monitoring of the propagation of abrupt traffic changes inside the ISP network. Various statistical measures e.g. volume, flow, entropy, ratio etc are available for profile generation. Let M and F be the random vectors compose of m measures $m_1, m_2, \ldots, m_m$ used for attacks detection and n flows $f_1, f_2, \ldots, f_n$ containing the incoming traffic to the server, respectively: $M = (m_1, m_2, \ldots, m_m)$, $F = (f_1, f_2, \ldots, f_n)$, where $f_i = (m_1^i, m_2^i, \ldots, m_m^i)$ is ith flow. Consider a random process $\{m_j^i(t),\ t = \omega\Delta,\ \omega \in N\}$, where Δ is a constant time interval, N is the set of positive integers, and for each t, $m_j^i(t)$ is a random variables. $1 \leq \omega \leq l$, l is the number of time intervals. Here $m_j^i(t)$represents the value of m_j in flow i in $\{t - \Delta, t\}$ time duration. These relations can be written in matrix form as follows:

$$Z(t) = \begin{pmatrix} m_1^1(t) & m_1^i(t) & \cdots & m_1^n \\ m_j^1(t) & m_j^i(t) & \cdots & m_j^n \\ \vdots & \vdots & & \vdots \\ m_m^1(t) & m_m^i(t) & \cdots & m_m^n(t) \end{pmatrix} \tag{11}$$

where, $Z(t)$ contains values of different measures used in $\{t - \Delta, t\}$. $m_j(t)$ represent total value of jth measure during $\{t - \Delta, t\}$ time. $m_j(t)$ can be calculated as follows:

$$m_j(t) = m_j^1(t) + m_j^2(t) + \cdots + m_j^i(t) + \cdots + m_j^n(t) \tag{12}$$

where $1 \leq i \leq n$, n is the number of flows. $1 \leq j \leq m$, m is the number of measures. Normal traffic value of jth measures can be calculated using following equation:

$$\overset{*}{m}_j(t) = \frac{1}{l}\sum_{\omega=1}^{l} m_j(t) \tag{13}$$

where $t = \omega\Delta$. Vector A can be used to represent normal traffic measures value: $A = (\overset{*}{m}_1(t), \overset{*}{m}_2(t), \ldots, \overset{*}{m}_m(t))$. To detect the attack, the value of jth traffic measure

$m_j(t)$ is calculated in time window Δ continuously; whenever there is appreciable deviation from $\overset{*}{m}_j(t)$, anomalous behaviors could be determined. Depending on the measures selected to use or network conditions, following events are defined to determine anomalous system behaviors:

$$m_j(t) - \overset{*}{m}_j(t) > \xi_j^{upper} \tag{14}$$

$$m_j(t) - \overset{*}{m}_j(t) < \xi_j^{lower} \tag{15}$$

where ξ_j^{upper} and ξ_j^{lower} represent value of upper and lower bound of the threshold for jth measure, respectively. ξ_j^{upper} and ξ_j^{lower} can be set as follows:

$$\xi_j^{upper} = r_j^{upper} * \sigma_j \tag{16}$$

$$\xi_j^{lower} = r_j^{lower} * \sigma_j \tag{17}$$

where σ_j represent value of standard deviation for jth measure. r_j^{upper} and r_j^{lower} represent value of tolerance factor to calculate upper and lower bound of the threshold for jth measure, respectively. Effectiveness of an anomaly based detection system highly depends on accuracy of threshold value settings. Inaccurate threshold values cause a large number of false positives and false negatives. Therefore, various simulations are performed using different value of tolerance factors. The choice of tolerance factors varies for different network conditions. Values of tolerance factors also depend on the composition of the normal traffic and the desired degree of the ability to control a DDoS attack. Then, trade-off between detection and false positive rate provides guidelines for selecting value of tolerance factor r_j for jth traffic measure for a particular simulation environment.

5.2 Entropy Based DDoS Detection

Here, we will discuss propose detection system that is part of access router or can belong to separate unit that interact with access router to detect attack traffic. It makes use of analytical model given in the previous section. Entropy based DDoS scheme [35] is used to construct profile of the traffic normally seen in the network, and identify anomalies whenever traffic goes out of profile. A metric that captures the degree of dispersal or concentration of a distribution is sample entropy. Sample entropy $H(X)$ is

$$H(X) = -\sum_{i=1}^{N} p_i \log_2(p_i) \tag{18}$$

where p_i is n_i/S. Here n_i represent total number of bytes arrivals for a flow i in $\{t - \Delta, t\}$ and $S = \sum_{i=1}^{N} n_i$, $i = 1, 2, \ldots, N$. The value of sample entropy lies in the range 0–$\log_2 N$. To detect the attack, the value of $H_c(X)$ is calculated in time window Δ continuously; whenever there is appreciable deviation from $X_n(X)$, various types of DDoS attacks are detected. $H_c(X)$, and $X_n(X)$ gives Entropy at the time of detection of attack and Entropy value for normal profile respectively.

6 Experimental Setup

In this section, we evaluate our proposed scheme using simulations. The simulations are carried out using NS2 network simulator. We show that false positives and false negatives (or various error rates) triggered by our scheme are very less. This implies that profiles built are reasonably stable and are able to predict number of zombies correctly.

6.1 Simulation Environment

Real-world Internet type topologies generated using Transit-Stub model of GT-ITM topology generator are used to test our proposed scheme, where transit domains are treated as different Internet Service Provider (ISP) networks i.e. Autonomous Systems (AS). For simulations, we use ISP level topology, which contains four transit domains with each domain containing twelve transit nodes i.e. transit routers. All the four transit domains have two peer links at transit nodes with adjacent transit domains. Remaining ten transit nodes are connected to ten stub domain, one stub domain per transit node. Stub domains are used to connect transit domains with customer domains, as each stub domain contains a customer domain with ten legitimate client machines. Total 400 legitimate client machines are used to generate background traffic. One FTP server is used to provide service to the clients. All FTP requests are originated randomly from different nodes. Total machines generating attack traffic range between 10 to 100. Transit domain four contains the server machine to be attacked by zombie machines. The legitimate clients are TCP agents that request files of size 1 Mbps with request inter-arrival times drawn from a Poisson distribution. The attackers are modeled by UDP agents. A UDP connection is used instead of a TCP one because in a practical attack flow, the attacker would normally never follow the basic rules of TCP, i.e. waiting for ACK packets before the next window of outstanding packets can be sent, etc. In our simulation experiments, attack traffic rate is fixed to 25 Mbps in total; and, mean attack rate per zombie varies from 0.25 Mbps to 2.5 Mbps as total number of zombie machines range between 10 and 100 to generate attack traffic. The simulations are repeated and different attack scenarios are generated by varying total number of zombie machines and at fixed attack strength. In our experiments, the monitoring time window was set to 200 ms. Total false positive alarms are minimum with high detection rate using this value of monitoring window.

7 Model Development and Performance Analysis

In this section, we describe our experiments to study the strength of various regression models for predicting number of zombies involved in a DDoS attack using isotonic regression. In order to predict number of zombies from deviation ($H_c - H_n$) in entropy value, simulation experiments are done at the same attack strength 25 Mbps

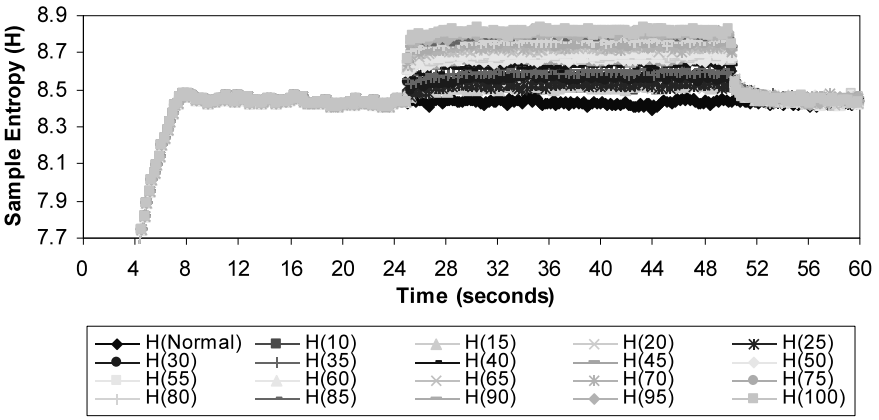


Fig. 1 Entropy variation with varied number of zombies

Table 1 Deviation in entropy with actual number of zombies

Actual number of zombies (Y)	Deviation in entropy (X) (H_c–H_n)
10	0.045
15	0.046
20	0.048
25	0.050
30	0.068
35	0.087
40	0.099
45	0.111
50	0.121
55	0.130
60	0.139
65	0.148
70	0.157
75	0.163
80	0.170

in total and varying number of zombies from 10–100 with increment of 5 zombies i.e. mean attack rate per zombie from 0.25 Mbps–2.5 Mbps.

Figure 1 shows entropy variation with 10–100 numbers of zombies at same attack strength in total of 25 Mbps. Table 1 represents deviation in entropy with actual number of zombies. Isotonic regression model is developed using number of zombies (Y) and deviation (H_c–H_n) in entropy value as discussed in Table 1 to fit the regression equation. Coefficients of regression equations are determined through a process of curve fitting. The main objective in the process of the curve fitting is to minimize the error between the actual number of zombies and the predicted number of zombies.

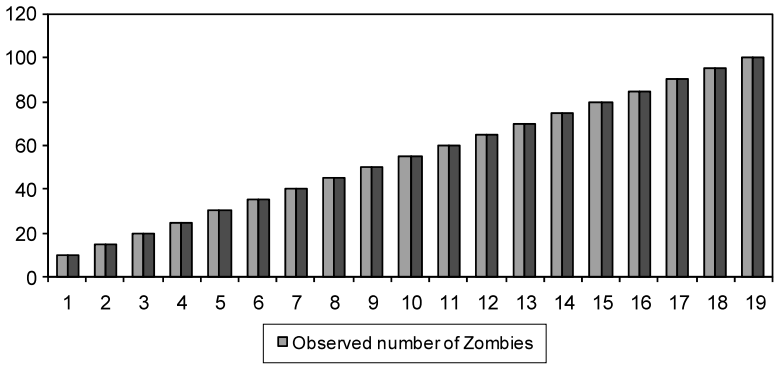


Fig. 2 Comparison between actual number of zombies and predicted number of zombies using isotonic regression model

8 Result and Discussion

We have developed isotonic regression model as discussed in Sect. 7. Various performance measures are used to check the accuracy of this model. Number of zombies can be computed and compared with actual number of zombies using proposed regression model. The comparison between actual number of zombies and predicted number of zombies using isotonic regression model is depicted in Fig. 2.

To represent false positive (falsely predicted normal clients as zombies) and false negative (zombies are identified as normal client) we plot residual error. Positive cycle of residual error curve represents false positive, while negative cycle represents false negative. As our model predicts number of zombies 100 % correctly, value of residual error is zero. Table 2 shows residual error for isotonic regression model. Table 3 shows values of various performance measures.

It can be inferred from Table 3 that for isotonic regression model values of R^2, CC, SSE, MSE, RMSE, NMSE, η, MAE are 1, 1, 0, 0, 0, 0, 1 and 1 respectively. Hence number of zombies predicted by this model is 100 % similar to the observed number of the zombies.

8.1 Comparison Between Isotonic Regression and Other Regression Models

Here performance between Isotonic and other regression models i.e. linear, polynomial, logarithmic, exponential, power etc. is compared to predict number of zombies involve in a DDoS attack. Figure 3 shows comparison between predicted number of zombies using Isotonic and other regression models. Similarly, Tables 4 and 5 represent comparison of residual errors and various performance measures for isotonic and other regression models.

Table 2 Summary of residual error for isotonic regression model

(X) Entropy variation	(Y) Number of zombies	Residual error
0.045	10	0.0
0.046	15	0.0
0.048	20	0.0
0.050	25	0.0
0.068	30	0.0
0.087	35	0.0
0.099	40	0.0
0.111	45	0.0
0.121	50	0.0
0.130	55	0.0
0.139	60	0.0
0.148	65	0.0
0.157	70	0.0
0.163	75	0.0
0.170	80	0.0
0.176	85	0.0
0.182	90	0.0
0.189	95	0.0
0.192	100	0.0

Table 3 Values of various performance measures

R^2	1
CC	1
SSE	0
MSE	0
RMSE	0
NMSE	0
η	1
MAE	1

As described in Sect. 4, coefficient of determination (R^2) is a descriptive measure of the strength of the regression relationship, a measure how well the regression line fit to the data. R^2 is the proportion of variance in dependent variable which can be predicted from independent variable and CC is its square root. The Nash–Sutcliffe efficiency index is a widely used and potentially reliable statistic for assessing the goodness of fit of models. Essentially, the closer the model efficiency is to 1, the more accurate the model is. On the other hand, values of SSE, MSE and NMSE quantify the error in the prediction using various regression models. Therefore, when comparing various regression models, model is selected with highest

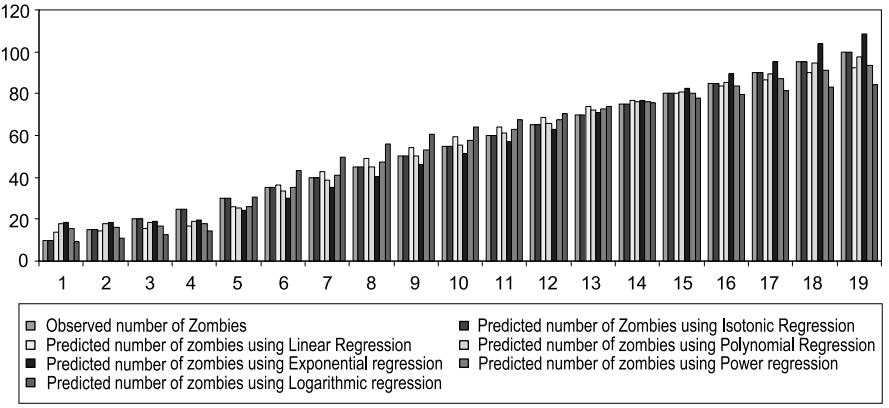


Fig. 3 Comparison between predicted number of zombies using Isotonic and other regression models

Table 4 Comparison of residual errors for isotonic and other regression models

(*X*) Entropy variation	(*Y*) Number of zombies	Residual error					
		Linear regression	Polynomial regression	Logarithmic regression	Power regression	Exponential regression	Isotonic regression
0.045	10	4.07	7.69	−0.53	5.62	8.27	0.0
0.046	15	−0.31	3.06	−4.22	1.12	3.53	0.0
0.048	20	−4.46	−1.43	−7.45	−3.19	−1.10	0.0
0.050	25	−8.39	−5.78	−10.33	−7.31	−5.64	0.0
0.068	30	−3.91	−4.46	0.43	−4.22	−5.97	0.0
0.087	35	1.14	−1.68	8.11	−0.08	−4.80	0.0
0.099	40	2.53	−1.15	9.81	0.99	−5.06	0.0
0.111	45	4.17	0.08	10.96	2.49	−4.36	0.0
0.121	50	4.51	0.45	10.42	2.84	−4.10	0.0
0.130	55	4.22	0.43	9.05	2.63	−3.91	0.0
0.139	60	4.12	0.88	7.58	2.71	−2.88	0.0
0.148	65	3.51	1.00	5.55	2.33	−1.87	0.0
0.157	70	3.75	2.37	3.88	2.91	1.13	0.0
0.163	75	1.94	1.40	0.81	1.35	1.49	0.0
0.170	80	0.23	0.67	−2.27	−0.07	2.43	0.0
0.176	85	−1.17	0.50	−5.25	−1.11	4.50	0.0
0.182	90	−3.40	−0.70	−8.75	−3.05	5.31	0.0
0.189	95	−4.75	−0.56	−11.84	−3.98	8.57	0.0
0.192	100	−7.80	−2.76	−15.85	−6.79	8.27	0.0

Table 5 Comparison of various performance measures for Isotonic and other regression models

	Isotonic	Linear	Polynomial	Logarithmic	Power	Exponential
R^2	1	0.98	0.99	0.91	0.95	0.92
CC	1	0.99	0.99	0.95	0.99	0.99
SSE	0	328.88	146.88	1257.90	231.85	460.81
MSE	0	17.31	7.73	66.21	12.20	24.25
RMSE	0	4.16	2.78	8.14	3.49	4.92
NMSE	0	0.62	0.27	2.35	0.43	0.86
η	1	0.98	0.99	0.91	0.98	0.97
MAE	1	0.85	0.92	0.70	0.88	0.82

value of coefficient of determination, coefficient of correlation and Nash–Sutcliffe efficiency index and lowest values of SSE, MSE and NMSE. Accordingly, it can be seen from Table 5 that isotonic regression model has highest value of coefficient of determination, coefficient of correlation and Nash–Sutcliffe efficiency index and lowest values of SSE, MSE and NMSE. Thus, it can be concluded that it performs better than other models. Figure 3 and Table 4 also show supremacy of isotonic regression model in predicting number of zombies in a DDoS attack.

9 Conclusion and Future Work

This chapter investigates suitability of isotonic regression model to predict number of zombies involved in a flooding DDoS attack from deviation in sample entropy. For evaluating performance of isotonic regression model, we have calculated various statistical performance measures. Based on the statistical measures, we found that isotonic regression based model performs better than any other model, as it can predict number of zombies in a DDoS attack with 100 % efficiency. However, simulation results are promising, experimental study using a real time test bed can strongly validate our claim.

References

1. Gupta BB, Misra M, Joshi RC (2008) An ISP level solution to combat DDoS attacks using combined statistical based approach. Int J Inf Assur Secur 3(2):102–110
2. Gupta BB, Joshi RC, Misra M (2009) Defending against distributed denial of service attacks: issues and challenges. Inf Secur J 18(5):224–247
3. Gupta BB, Joshi RC, Misra M (2009) Dynamic and auto responsive solution for distributed denial-of-service attacks detection in ISP network. Int J Comput Theory Eng 1(1):71–80
4. Wu WB, Woodroofe M, Mentz G (2009) Isotonic regression: another look at the changepoint problem. Biometrika 88(3):793–804
5. Barlow RE, Brunk HD (1972) The isotonic regression problem and its dual. J Am Stat Assoc 67(337):140–147

6. Moore D, Shannon C, Brown DJ, Voelker G, Savage S (2006) Inferring internet denial-of-service activity. ACM Trans Comput Syst 24(2):115–139
7. GT-ITM Traffic Generator documentation and tool. Available at http://www.cc.gatech.edu/fac/EllenLegura/graphs.html
8. Documentation NS. Available at http://www.isi.edu/nsnam/ns
9. Ferguson P, Senie D (1998) Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing. In: RFC 2267, the Internet Engineering Task Force (IETF)
10. Peng T, Leckie C, Ramamohanarao K (2003) Protection from distributed denial of service attack using history-based IP filtering. In: Proceedings of ICC 2003, USA, pp 482–486
11. Molsa J (2005) Mitigating denial of service attacks: a tutorial. J Comput Secur 13:807–837
12. Geng X, Whinston AB (2000) Defeating distributed denial of service attacks. IEEE IT Prof 2(4):36–41
13. Paxson V (1999) Bro: a system for detecting network intruders in real-time. Int J Comput Telecommun Netw 31(24):2435–2463
14. Roesch M (1999) Snort-lightweight intrusion detection for networks. In: Proceedings of the USENIX systems administration conference, LISA '99, pp 229–238
15. Gil TM, Poletto M (2001) Multops: a data-structure for bandwidth attack detection. In: Proceedings of the 10th USENIX security symposium, Washington, DC, USA, pp 23–38
16. Blazek RB, Kim H, Rozovskii B, Tartakovsky A (2001) A novel approach to detection of denial-of-service attacks via adaptive sequential and batch sequential change-point detection methods. In: Proceedings of IEEE systems, man and cybernetics information assurance workshop, pp 220–226
17. Cheng CM, Kung HT, Tan KS (2002) Use of spectral analysis in defense against DoS attacks. In: Proceedings of IEEE GLOBECOM 2002, Taipei, Taiwan, pp 2143–2148
18. Lee W, Stolfo SJ, Mok KW (1999) A data mining framework for building intrusion detection models. In: Proceedings of the 1999 IEEE symposium on security and privacy, Oakland, CA, pp 120–132
19. Huang Y, Pullen JM (2001) Countering Denial of Service attacks using congestion triggered packet sampling and filtering. In: Proceedings of the 10th international conference on computer communications and networks, Scottsdale, Arizona, pp 490–494
20. Mirkovic J, Prier G, Reiher P (2002) Attacking DDoS at the source. In: Proceedings of ICNP-2002, Paris, France, pp 312–321
21. Bencsath B, Vajda I (2004) Protection against DDoS attacks based on traffic level measurements. In: Proceedings of the western simulation multi conference, San Diego, California, pp 22–28
22. Chen Y, Hwang K, Ku W (2007) Collaborative detection of DDoS attacks over multiple network domains. IEEE Trans Parallel Distrib Syst, TPDS-0228-0806 (12)
23. Feinstein L, Schnackenberg D, Balupari R, Kindred D (2003) Statistical approaches to DDoS attack detection and response. In: Proceedings of DISCEX'03, Washington, DC, USA, pp 303–314
24. Savage S, Wetherall D, Karlin A, Anderson T (2000) Practical network support for IP traceback. In: Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, pp 295–306
25. Snoeren AC, Partridge C, Sanchez LA, Jones CE, Tchakountio F, Kent ST, Strayer WT (2001) Hash-based IP traceback. In: Proceedings of ACM SIGCOMM 2001, San Diego, CA, USA, pp 3–14
26. Darmohray T, Oliver R Hot spares for DDoS attacks. http://www.usenix.org/publications/login/2000-7/apropos.html
27. Mahajan R, Bellovin SM, Floyd S, Ioannidis J, Paxson V, Shenker S (2002) Controlling high bandwidth aggregates in the network. Comput Commun Rev 32(3):62–73
28. Lau F, Rubin SH, Smith MH, Trajkovic L (2000) Distributed denial of service attacks. In: Proceedings of IEEE international conference on systems, man, and cybernetics, pp 2275–2280

29. Floyd S, Jacobson V (1993) Random early detection gateways for congestion avoidance. IEEE/ACM Trans Netw 1(4):397–413
30. Demers A, Keshav S, Shenker S (1990) Analysis and simulation of a fair queuing algorithm. J Internetworking Res Exp 1(1):3–26
31. Khattab SM, Sangpachatanaruk C, Melhem R, Mosse D, Znati T (2003) Proactive server roaming for mitigating denial-of-service attacks. In: Proceedings of international conference on information technology: research and education, ITRE2003, pp 286–290
32. Kumar K, Joshi RC, Singh K (2007) Predicting number of attackers using regression analysis. In: Proceedings of IEEE international conference on information and communication technology, Bangladesh, pp 319–322
33. Lindley DV (1987) Regression and correlation analysis. New Palgrave, A Dict Econ 4:120–123
34. Freedman DA (2005) Statistical models: theory and practice. Cambridge University Press, Cambridge
35. Shannon CE (2001) A mathematical theory of communication. Mob Comput Commun Rev 5(1):3–55

Developing a Hybrid Framework for a Web-Page Recommender System

Vasileios Anastopoulos, Panagiotis Karampelas, and Reda Alhajj

Abstract Recommender systems nowadays tend to become a necessity against information and product overloading. They aim to facilitate users browsing the World Wide Web by suggesting relevant products, websites or services according to users' preferences. In this paper we present a hybrid framework that analyzes Web logs using social network analysis and data mining techniques, to extract useful information about users browsing patterns. Based on the identified results the framework builds a recommendation engine that is integrated in the Web browser of the user. A case study based on real data from an organization of 250 employees is finally presented using the system prototype which was constructed based on the proposed framework.

Keywords Recommender system · Social network · Data mining · Association rules · System prototype · Hybrid framework

1 Introduction

It is widely accepted nowadays that the World Wide Web has become the main source of information for practically any information someone would like to get. The Web is developing into the main source for everything that could come to mind and the user group is growing exponentially increasing to include mostly users with limited background and capabilities. This abundance of information, together with the advertisements and in combination with the various hyperlinks inherent to the Web pages, has increased cognitive load of a user with information that most of the times is irrelevant to his/her interest on a topic. Thus, finding information has

V. Anastopoulos (✉) · P. Karampelas
Hellenic American University, Manchester, NH, USA
e-mail: vasanasto@gmail.com

P. Karampelas
e-mail: pkarampelas@hauniv.us

R. Alhajj
Dept. of Computer Science, University of Calgary, Calgary, Canada
e-mail: alhajj@ucalgary.ca

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_9,

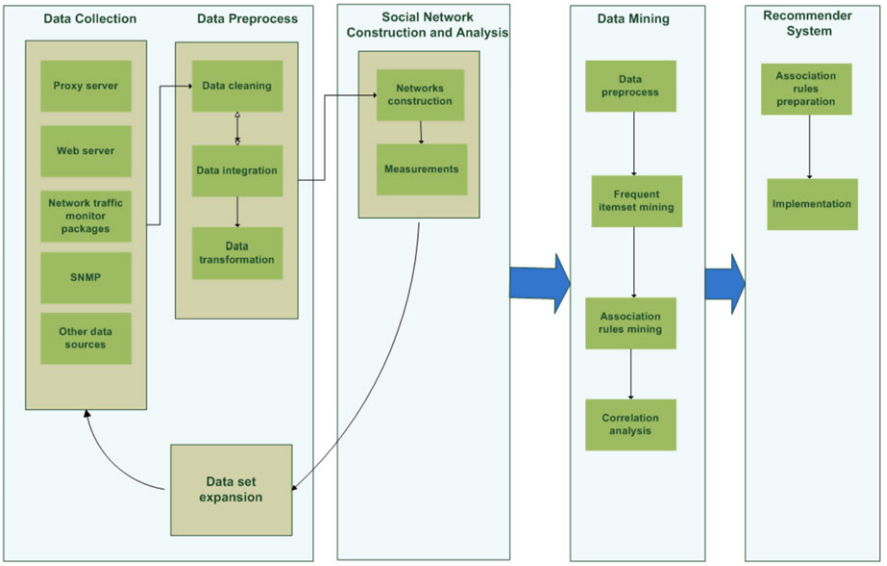


Fig. 1 Proposed framework block diagram

changed into a time consuming, effort demanding and sometimes disturbing task for both novice and experienced users. To facilitate users in this quest, recommendation systems are implemented as an intermediate service between the user and the provider of the information, aiming to recommend items of his/her interest based on the user's preferences. This approach increases user satisfaction and provides them a browsing experience adjusted to their own preferences and needs. Recommendation systems usually draw data from the existing users of a website or service. These data are commonly recorded into the log files of Web and proxy servers resulting into data sources that contain record entries that store the time and the URL of the Web pages accessed by each user of the network. From these data, valuable information can be extracted about each user's preferences, increasing the accuracy and quality of the recommendations presented. This paper proposes a hybrid framework that analyzes Web logs using social network analysis with data mining techniques, to increase the accuracy and the quality of Web page recommender systems. Fortunately, users could be classified into groups such that each group has similar preferences. Thus, identifying the most appropriate group that best matches a given user would lead to more accurate recommendation. The work presented in this paper is divided into the following four main phases, a) data collection, b) social network analysis, c) data mining, and d) implementation of the recommender system, as depicted in Fig. 1.

Data is collected from various sources which may be open or proprietary and can provide information about the time and the URL of the Web pages a user has requested. A proxy server can be used for the collection of this information, especially within the boundaries of an organization, since all user requests are satisfied

through it. Web servers can also provide this data, with the drawback of logging only the requests of Web pages hosted on those specific servers. If no privacy and security issues are raised, network traffic monitor packages (ethereal, SNMP protocol, etc.) can collect the necessary data, monitoring specific servers or intermediary devices. The output of the data collection process is log files, possibly from different sources and in various formats. Depending on the data source they can be stored in flat files of database tables and will probably contain redundant data, due to automatic Web browser requests and the stateless HTTP protocol. These Web logs require pre-processing to remove unnecessary or private data and integrate them into one data set that contains only the necessary information in the appropriate format for the social network analysis. It is important to avoid the loss of useful data and retain the quality of the data set at the same time. This is accomplished by applying techniques [6] that preserve the requests that are necessary to form the picture of the user's usage, i.e., HTML files explicitly requested, thus, remove the implicit ones (graphics and scripts requested by the Web browser), that are irrelevant to the user's browsing behavior.

The data set is then represented as a social network of users and Web pages and various measurements are performed and analyzed to conclude on the network stability and cohesion, as well as on the importance of certain actors. The analysis is initially performed on a small data set, but in order to have an overall picture of the network dynamics it needs to be extended to include larger time periods that will include characteristic time periods e.g., a full work day or a work week. The process followed so far, is repeated until a sufficient amount of data has been analyzed and the conclusions on the network stability, the actors importance and patterns of usage are concrete.

The framework continues with the data mining process that again pre-processes the Web logs to transform them to the appropriate format for the application of data mining algorithms. The frequent item-sets mining results into frequent patterns of Web pages access and are used as input for the association rules mining; this consequently results in a set of associations between Web pages, based on which recommendations will be made by the recommender system. The construction of the recommender system starts with the preparation of the grouping of rules and the selection of the more accurate ones. Finally, the recommender system is implemented focusing on the recommendation engine which is a system prototype that runs completely on the client's browser, differentiating from the usual approach that places the recommendation engine at a Web or proxy server. The proposed framework benefits from the available Web log analysis methods to prepare the data for social network analysis and data mining. Social network analysis facilitates the identification of important Web pages and users, representing the usage of the pages from the users as a network of interaction between the users and the Web pages. For the generation of recommendations the memory-based approach of association rules mining is employed, as it can be applied efficiently on large data sets and the quality of the recommendations can be easily evaluated. The combination of different methods or approaches increases the complexity of the construction process, but significantly benefits from the advantages of each method applied which is the main objective of the proposed framework.

Social network analysis is usually performed on networks where people are the actors and by finding friends, communities or similarities among them, enhances the quality of the recommendations. The proposed framework differentiates from this approach, choosing Web pages as actors of the social network. These Web pages are used to construct social networks and then are analyzed in terms of importance, connectivity and "socialization" over the time. The users are also part of the approach as through their usage behavior the Web pages are implicitly connected forming social networks. The users influence Web page networks by adding the dynamic and evolutionary features to these networks. The information extracted from the analysis of both users and Web pages social networks leads to useful conclusions that are used to increase the quality of the recommendation engine.

The remainder of the paper is organized as follows. Section 2 provides a brief overview of related work on Web log pre-process approaches of mining Web page navigational patterns and the types of recommender systems that are common in literature. Section 3 describes in detail the proposed framework, the problems and methods of the data pre-process, the measurements and algorithms that are applied, as well as the process of constructing the recommendation engine. Following, in Sect. 4 the framework is applied on real usage data from an organization of 250 employees and results in a system prototype that is implemented as an extension of the Google Chrome Web browser. The framework is evaluated in Sect. 5, and the paper concludes in Sect. 6 with the conclusion and future work that could improve the performance and accuracy of the recommender system.

2 Related Work

The most important problem in collecting reliable usage data is caching, either from the users' browsers or proxy servers. This process is necessary when the objective is to minimize the traffic over the network and increase performance. As a result, Web server logs do not include requests that were satisfied by locally cached Web pages, or in the case of proxy server intermediation, all requests have the same identifier even though they correspond to various users. Cooley, Mobasher and Srivastava [6] confronted this problem with preprocessing of the logs, user identification and session identification.

Preprocessing of the logged data is necessary to remove records that are not actually relevant to the user browsing behavior. These records are HTTP requests that are implicitly made from the browser in order to complete the display of the Web page. As the HTTP protocol requires each request to be a separate session, a record log is created for each one. A common solution to this problem which was also employed in the proposed framework is to remove requests based on a list of suffixes, i.e. jpg, jpeg, gif, cgi, etc., or list of irrelevant file names. Depending on the type of data collected, the list of suffixes to be removed can vary.

User identification is another important task since requests of different users are logged in the proxy server as being made from the same IP address. This task is more

complicated and the proposed methods in the literature rely on the cooperation of the user or on heuristics [6]. The user's cooperation is usually achieved by requiring login first to a Web site that tracks the usage. Accepting cookies from a server is another form of user cooperation, as the user's browser will send the cookie with each new request, and thus by identifying the cookie the Web server actually can identify the user. There are however serious drawbacks in this approach since the user may delete stored cookies or be negative to login prior to Web browsing, as privacy is in most cases of primary concern. Heuristics are mostly based on the assumption that different operating systems or Web browsers at the same IP address indicate different users, but two users with the same IP address that use the same browser on the same operating system can be easily be considered as a single user. Another heuristic method is to combine the Web log with the site topology. A Web page that is not reachable through the links of the Web pages already accessed by the user can be assumed that was requested by another user having the same IP.

Session identification is usually applied in usage logs that cover long periods of time, since a user may visit the same Web page more than once during this period. Each time the user accesses the Web site it is considered a new session and the aim of the method, is to divide the Web pages the user has accessed to separate sessions. A common approach is to define a timeout, which in literature varies from ten minutes to two hours, after which it is assumed that the user starts a new session.

Coming to the construction of collaborative recommender systems, the most well-known approaches in literature are the use of memory-based and model-based algorithms [14]. Memory-based algorithms use the entire data set that corresponds to the items each user accessed and is represented as a user-item matrix. In order to generate recommendations the k-nearest-neighborhood or association rules algorithms can be applied. In k-nearest-neighborhood, the similarity between users in item rating or accessed Web pages is calculated in the user-item matrix and similar users form a proximity-based neighborhood. It is assumed that the items accessed by the user's neighbors will probably interest him/her and thus they are recommended.

Association rules are usually applied to "market basket" data, meaning that each user transaction has an ID and the items accessed. Usually they are mined using the Apriori or FP-growth algorithms. These algorithms initially generate frequent itemsets, which are patterns that appear often in the transactions and then associations between these sets of items are derived [7]. Association rules are interpreting as if a user accessed items A, then he/she will probably access items B ($A \Rightarrow B$). The strength of these rules is evaluated by their support and confidence. Association rules are used in the proposed framework to create recommendation, and they are presented in detail to the following sections.

The model-based collaborative filtering approach, aims to derive a model from the rating data that will be used in continuance to generate recommendations. This is achieved by applying machine learning algorithms, such as neural networks, Bayesian networks, clustering and latent semantic analysis [16]. Each of these methods is covered in detail in data mining literature and they are not discussed in this work since they are not related to the proposed framework. In recent literature, social network analysis algorithms are also combined with data mining, representing

the collaborative relationships as social networks. These networks are analyzed in order to understand the relationships, between users and items of their interest, the collaboration among users, how they change in time and their reflections on their preferences [16]. In [1], the authors propose a recommendation system that applies social network analysis to classify the users based on their ratings and their browsing behavior. Users that visited the same Web pages are considered similar and they are linked to create social networks, which are then analyzed to detect their habits. A new user entering the system is provided with recommendations according to the ratings collected by similar users. In a recommender system [13] that aims to understand the collaboration among users, its users are represented as social networks. The users are linked based on their exchange of information and then social network analysis measurements are applied to interpret the creation of those connections between them. Another recommender system for links in a social network is presented in [9]. The authors investigate the problem of link recommendation in Web log based social networks and describe an annotated graph-based representation for such networks. Structural features of individual vertices and joint features of the start and end points of a candidate link are analyzed to create recommendations for new links. Another recommender system for products in [17] benefits from link analysis. The data obtained, from the users past acquisitions, are projected into a bipartite network, where a connection between the two layers is created whenever a user buys a certain item. After a diffusion process, the system outputs a value for each item, indicating how close is that item to the target user and, thus recommends the higher valued ones. The role of explicit social relations in recommender systems, i.e., how user preferences or ratings correlate with those of their friends and how to use these correlations to increase the quality of recommendations, is presented in [8]. The authors propose an algorithmic framework that bases recommendations on the user's own preferences, the general acceptance of the target item and the opinions from his/her friends in the social network. Social networks analysis leads to better understanding of user behaviors and ratings, and under the assumption that friendship between users indicates common interests and preferences, alleviates the cold-start problem.

Based on social network analysis, new friends or new professional contacts can be recommended. Merging social network analysis and data mining has proved to increase the quality of the data and as a result the efficiency of the recommender systems.

3 Proposed Framework

3.1 Data Collection and Pre-process

The required data to work with the proposed framework are the Web pages that have been accessed by each user. These pages can be collected using the SNMP protocol, with applications that monitor the traffic on a network such as *tcpdump*,

Table 1 Sample data from the ISA proxy server logs

#Fields: c-ip	Date	Time	r-host	r-ip	GMT time
192.168.1.61	09/28/2010	10:22:22	static-0.farmville.com	192.168.8.10	09/28/10 10:22 AM
192.168.1.66	09/28/2010	10:22:22	www.google.gr	192.168.8.10	09/28/10 10:22 AM
192.168.1.66	09/28/2010	10:22:23	www.google.gr	66.249.92.104	09/28/10 10:22 AM
192.168.1.66	09/28/2010	10:22:23	clients1.google.gr	192.168.8.10	09/28/10 10:22 AM
192.168.1.66	09/28/2010	10:22:23	www.google.gr	66.249.92.104	09/28/10 10:22 AM
192.168.1.66	09/28/2010	10:22:23	clients1.google.gr	209.85.229.100	09/28/10 10:22 AM
192.168.1.59	09/28/2010	10:22:23	swupmf.adobe.com	192.168.8.10	09/28/10 10:22 AM
192.168.1.61	09/28/2010	10:22:25	fb-tc-2.farmville.com	204.236.227.41	09/28/10 10:22 AM
192.168.0.95	09/28/2010	10:22:25	live24.gr	192.168.8.10	09/28/10 10:22 AM
192.168.0.95	09/28/2010	10:22:25	live24.gr	192.168.8.10	09/28/10 10:22 AM
192.168.0.95	09/28/2010	10:22:26	live24.gr	70.84.186.50	09/28/10 10:22 AM

argus, *mrtg*, *ethereal* and other packages or logged information from Web and proxy servers. Initially, a data set is collected covering a small period of time, e.g., a few hours, and consequently this data set is extended to several days of network traffic.

Data pre-process is a necessary process to improve the quality of the data and as a consequence the results of link analysis and data mining [7]. The input in this stage is the previously mentioned data while the output is data sets containing only the necessary data for the link and data analysis processes to follow. The different steps of the data pre-process stage can be summarized to data cleaning, data integration, data transformation and data reduction. Data cleaning attempts to correct incomplete, noisy and inconsistent data. Data integration then merges data from various sources, since network traffic can be recorded into flat files and database tables, while transformation brings them to the appropriate format for analysis and mining to be performed by the software tools. The data is then reduced, replacing each Web page with a numeric value, making it is easier to handle and less demanding in storage.

The format of the data contained in Web logs is shown in Table 1, where a sample of real usage data is listed, captured by a proxy server; no pre-processing techniques have been applied yet. Whenever an HTTP request is made, either implicit or explicit, a new row is added. The columns that are necessary for the proposed framework are *c-ip*, the IP address of the requesting client, *r-host*, the Web page that was requested, and the dimension of time that is contained in *GMT-Time*, the time stamp of the request. The remaining columns are redundant and they are removed during the data integration and reduction, while the final step, data transformation, replaces each URL with a number resulting to the format shown on Table 2.

The main problem with the Web logs is that both proxy servers and browsers cache Web pages and that the Web browsers automatically request new content in order to complete the display of a Web page. This results in difficulties to identify the user and its behavior, in addition to the HTTP protocol that requires a separate

Table 2 Sample data from the *r-host* column

Web page	ID
www.farmville.com	1
www.google.gr	2
news.google.gr	3
swupmf.adobe.com	4
facebook.farmville.com	5
live24.gr	6
www.logitech.com	7
pmetrics.performancing.com	9
www.bbc.co.uk	11
www.contra.gr	12
lt.andomedia.com	13
english.aljazeera.net	14
www.adman.gr	1

connection for every requested file. There are processes available to overcome these problems presented at [6] aiming to user and session identification and including user path completion and formatting. The pre-processing to be performed depends on the actual data that are considered adequate in order to identify the user's behavior. If, for example, a user is accessing paintings on a Web site, the *jpg* or *gif* requests should not be removed but in all other cases they would be removed as automatic requests of the browser to complete the display of the Web page.

3.2 Social Network Construction and Analysis

The next stage of the proposed method is to construct the social networks of the Web pages and then their analysis is following. The social networks are composed of nodes and links. These nodes relate with other nodes through their links. The links can have a direction; the link from node A to B is different from node B to A. A 2-mode network is represented with an incidence matrix, where a value indicates the presence of a link. The value is a number if the link is weighted or 1 otherwise. This 2-mode network can then be folded to create two 1-mode networks, one for each dimension. To fold a network, it is first transposed to the desired dimension and then multiplied with the initial incidence matrix, resulting to the adjacency matrix.

In our framework, the construction of the 2-mode network begins with a small data set, a few hours of network traffic. These relational data are used to create the incidence matrix, which is a $|V|$ by $|E|$ array. The $|V|$ array is the host IP addresses and the $|E|$ array the Web pages requested, when a host i requests a Web page j the weight of the link is added to cell ij.

The two-dimension incidence matrices will be folded into both dimensions to form two 1-mode networks, with the respective adjacency matrices. The result of

the $|V| \times |E|$ folding will be the $|V| \times |V|$ and $|E| \times |E|$ arrays, where each cell contains the weight between v_i and v_j, e_i and e_j, respectively.

The analysis of the two 1-mode networks aims to the identification of important nodes in each network, meaning important users and Web sites. To measure importance the degree, closeness, eigenvector and betweenness centrality will be measured.

The *degree centrality* is the number of links that a node has and is distinguished into in an out degree, when the links are directed to or from the node, respectively. In our case, the constructed network is undirected, so there is no need to distinguish the in from the out degree; the total degree centrality of the nodes will be measured. Let $G = (V, E)$ be the graph representation of a square network and a node v. The Total Degree Centrality of node $v = deg/2 * (|V| - 1)$, where $deg = card\{v \in V | (v, u) \in E \vee (u, y) \in E\}$ [15]. A node with high degree centrality is well connected node and can potentially directly influence many other nodes [3].

Closeness centrality is the average geodesic distance of a node from all other nodes in the network, where geodesic distance is the length of the shortest path between two nodes. Let $G = (V, E)$ be the graph representation of a square network, then the closeness centrality of a node $v \in V$ is $v = (|V| - 1)/dist$, where $dist = \sum d_G(v, i)$, $i \in V$, if every node is reachable from v and $v = |V|$ if some node is not reachable from v ([11] as cited in [5]). The closest a node is to others, the fastest its access to information and influence to others [10].

Another measurement that is used to identify important nodes is *betweenness centrality*, which is defined for a node v, as the percentage of shortest paths, between node pairs, that pass through v. Let $G = (V, E)$ be the graph representation of a symmetric network. Let $n = |V|$ and a node $v \in V$. For $(u, w) \in V \times V$, let $n_G(u, w)$ be the number of geodesics in G from u to w. If $(u, w) \in E$, then set $n_G(u, w) = 1$. Now, let $S = (u, w)V \times V | d_G(u, w) = d(u, y) + d(v, w)$ and let $between = \sum(n_G(u, v) \times n_G(v, w))/n_G(u, w)$, $(u, w \in S)$, then the betweenness centrality of node $v = between/((n - 1)(n - 2)/2)$ ([11], as cited in [5]). A node with high betweenness is important because it connects many nodes and a possible removal would affect the network.

The last node level measurement that is applied is the *eigenvector centrality*. It is a measure of the node's connections with other highly connected nodes. It calculates the eigenvector of the largest positive eigenvalue of the adjacency matrix representation of the square network. To compute the eigenvalues and vectors a Jacobi method is used ([2], as cited in [5]). The nodes with high eigenvector centrality can mobilize other important nodes [10].

Apart from the node level measurements it is important to analyze the networks, from a network level perspective. The measurements that are applied in the proposed framework are density, fragmentation, component count, isolate count. These measurements are very useful as they describe the network as a whole [5], which in combination with the node level measurements provide us comprehensive information about the networks' cohesion.

Fragmentation measures the proportion of the network's nodes that are disconnected. Let an undirected network, $G = (V, E)$ with $n = |V|$ and s_k be the

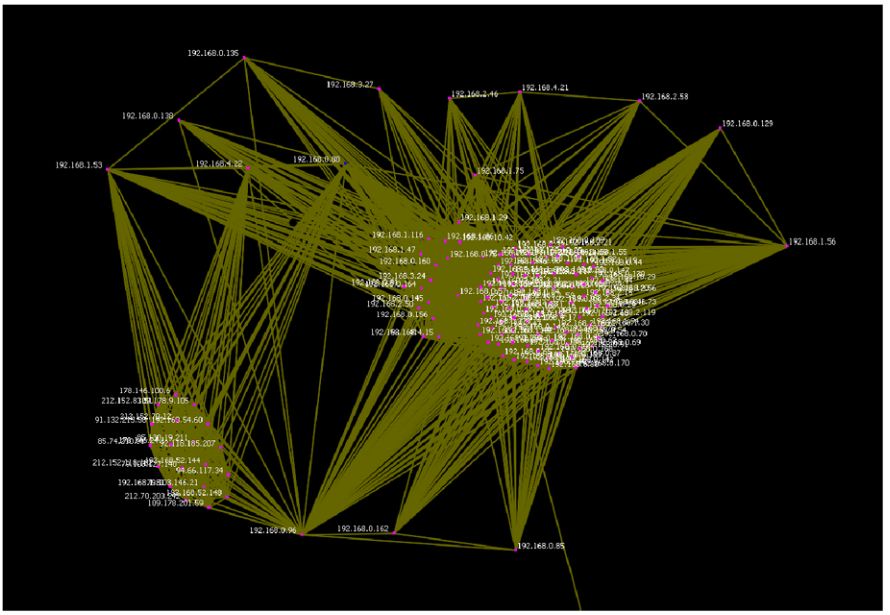


Fig. 2 Visualization of the host IP address 1-mode network

number of nodes in the kth component of G, $1 \leq k \leq n$, then $Fragmentation = 1 - (\sum s_k(s_k - 1))/n(n-1)$ ([3], as cited in [5]).

Density is the ration of the number of links, existing on a network, versus the maximum possible ones. For a network with adjacency matrix M and dimensions $m \times n$, the density is calculated by the form $Density = sum(M)/(m * (m-1))$, if the network is unimodal and $Density = sum(M)/(m * n)$, if the network is bimodal the $Density = sum(M)/(m * n)$ ([15], as cited in [5]).

In addition to the above-mentioned measurements, visualization of the social networks can also assist in the identification of important actors. In Fig. 2 an 1-mode network is presented, where the nodes are the host IP addresses and the links between them indicate how they are connected to each other. Even though the number of links and actors is rather big, and thus the network very complex, nodes with high centrality can be identified, as well observations on the networks structure can be drawn. Most social network analysis tools support isolating part of the network or zooming into it, facilitating their detailed analysis.

Following the identification of important nodes in the networks, the proposed framework continues with the removal of the top valued ones. The nodes are sorted with descending order and we start removing the top nodes one-by-one, repeating the measurements after each removal. This process is repeated for each measurement and it aims to observe how the network is affected from the removal of each node. The removal of a node is an exogenous impact, a shock for the network, whose results to the network's dynamics and cohesion are observed. A network may remain stable, which means that the links between the nodes do not change, or

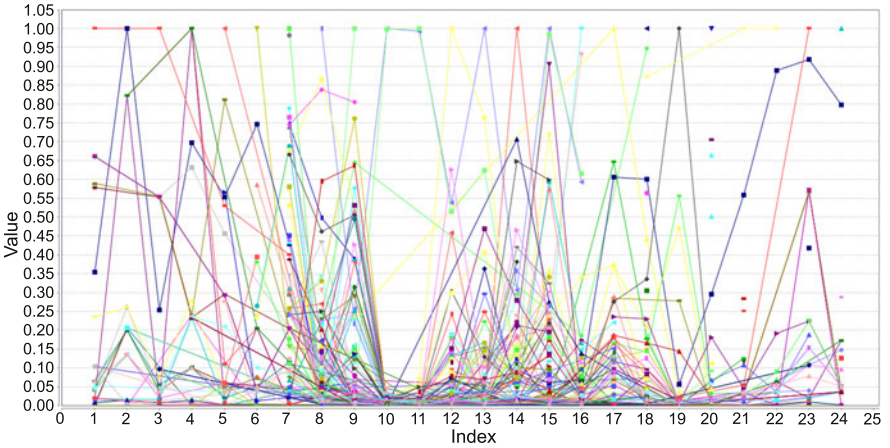


Fig. 3 Total degree centrality of host IP 1-mode network, 24 hours

mutate, meaning that an evolutionary process is initiated [12], or its cohesion may change by increasing the network's fragmentation and components [4].

The next phase in the proposed framework is to extend the data collection to the period of one day. This data set is divided to characteristic time periods and is further analyzed, repeating the measurements previously presented.

The data set is divided to twenty-four subsets, one for each hour of the day. The node and network level measurements are calculated for each network and the results are compared in the dimension of time to identify its dynamics, whether it is stable or not, whether it remains coherent or not. Drastic changes are obvious in social networks, whilst small are difficult to detect. This makes Cumulative Sum (CUSUM) Control charts suitable for social networks, as they perform well in detecting small changes over time and also provide detection of the point the change occurred [12]. The CUSUM control chart compares sequentially the statistic C_t against a decision interval until $C_t > A'$. Since one is not interested in concluding that the network process is unchanged, the cumulative statistic is $C_t^+ = \max\{0, Z_t - k + C_{t-1}^+\}$. The Exponentially-weighted Moving Average (EWMA) control chart and moving window analysis are also applicable, but in the proposed framework only CUSUM control charts are generated.

The analysis of a network on the time domain, may lead to erroneous conclusions because of periodicity. For example, a company meeting scheduled to take place at a specific day and time, every week, could mislead the analysis to identify a shock at that network. In the proposed framework the process to identify and handle periodicity analyzed in [12], is used, applying spectral analysis to the network's data on the dimension of time. For example, in Fig. 3 the measurements performed on the host IP 1-mode network for the time period of one day are presented. The x-axis shows the time domain and the y-axis the total degree centrality measured. Each date series corresponds to one node (host IP address). Some nodes' measurement oscillates during the day, though most of them remain stable. An interesting observation

is that the curve identified between 10:00 and 11:00 could possibly be interpreted as a shock of the network. Spectral analysis could possibly reveal that this is a periodic event e.g., a meeting or systems maintenance, and thus avoid erroneous conclusions.

In continuance, the same data set is divided into two subsets, working hours and not-working hours, and the same analysis is applied. Finally, the same process is followed after dividing the data into four subsets, the working hours split into two and the non-working hours split into two other subsets. To conclude, the data collected are extended to six days and the same procedure is repeated. The output of this analysis helps us determine the cohesion and stability of the data. Patterns or specific time periods might need to be taken into account while mining for the association rules.

3.3 Data Mining

In this section the data mining techniques that are applied to the dataset are described. Depending on the results of the social network analysis, on the network's stability or patterns of usage, the logs can be divided by day or by specific time periods and the data mining process can be repeated to each one of these subsets. The process starts with the preprocessing of the data, continues with the identification of the frequent item sets and then the association rules mining algorithms are applied.

The transactional data, as previously prepared and used so far are in a multi-instance format as shown in Table 3(a). In order to apply the data mining algorithms they need to be transformed to the single-instance format of Table 3(a). The multi-instance data set is the output of the pre-processing phase in the previous stage and is appropriate for the construction of the social networks. For the data mining algorithms the data should be transformed to single-instance transactional data. In this form, each row has an ID which in this case is the requesting host IP address, in the left column, and in the right column the Web pages accessed from this host are listed. This format is common in "market basket analysis" where products or items that are frequently purchased together are mined.

An itemset in this case is a set of Web pages. An itemset containing k items is a k-itemset, which has a support count that equals to the number of its occurrences in the transactions data set. When an itemset satisfies a minimum support count threshold, it is a frequent itemset, denoted by Lk. The mining of the frequent itemsets can be performed applying Apriori or FP-Growth algorithms. Apriori is a seminal algorithm that scans the data set to find frequent 1-itemsets and then joins them to generate candidate 2-itemsets. These candidate itemsets are evaluated by scanning again the data set and the iterations continue finding $(k + 1)$-itemsets from previously known k-itemsets. The drawbacks of this process are that a huge amount of candidate sets may be generated and the cost in terms of the repeated transactions while scanning the database [7, 16].

In the proposed framework, the FP-Growth algorithm was selected, as it is faster than Apriori and is more appropriate for large data sets. The algorithm applies a

Table 3 Instances and transactional data

(a) Multi-instance vs. single-instance

IP	Web page
IP B	40
IP C	138
IP C	139
IP C	140
IP D	138
IP D	139
IP D	145

(b) Transactional data

IP	Web page
IP B	40
IP C	138,139,140
IP D	138,139,145

divide-and-conquer approach which consists of two steps, the FP-tree construction and the mining of the frequent itemset. To construct the FP- tree, a "null" root node is created and then the data set is scanned to obtain a list of frequent items. These are ordered in descending order based on their support. Using this order, the items in each transaction of the data set are reordered, while each node n in the FP-tree represents a unique itemset X. All the nodes have a counter indicating the transactions that share the node, except for the root node. The algorithm scans the items in each transaction, it searches the already existing nodes in FP-tree and if a representative node exists the counter of the node is incremented by 1, else, a new node is created. The support of each item is stored in a header table, while the same table is used for each item to point to its occurrences in the tree. This way the problem of mining large data sets for frequent patterns, has transformed to the mining of the FP-tree. The FP-tree mining starts from each frequent pattern of length-1 (initial suffix), constructing its conditional pattern base (the set of prefix paths in the FP-tree co-occurring with the suffix pattern), then constructs its conditional FP-tree and mines recursively this tree. The pattern growth is achieved concatenating the suffix pattern with the frequent patterns generated from the conditional FP-tree [7]. In Table 4, a sample of frequent itemsets as the result of the application of FP-Growth algorithm is presented. Interpreting these data we conclude that Web pages 2 and 20 are both requested by users, with a frequency of 6.7 %, in the mined data set.

The above-mentioned algorithms provide us with a set of frequent itemsets, which will be used as input for the association rules mining algorithm. An association rule is an expression $A \Rightarrow B$, where A and B are itemsets of Web pages. This rule is translated in *if a host requested the Web pages of A, then he/she will also request the Web pages of B itemset*. The support of rule is represented by $support(A \Rightarrow B) = support(A \cup B)$ and it is defined as the percentage of the transactions in which this association rule appears, while the confidence of the association rule, $confidence(A \Rightarrow B) = support(A \cup B)/support(A)$, is again a percentage that indicates the conditional probability that a transaction containing A will also contain B. The higher the support value and confidence is, the stronger the rule is. Both measurements have to satisfy the thresholds that are set previously from the analyst [16].

Table 4 Sample frequent itemsets

Size	Support count	Items			
2	0.280	2	3		
2	0.082	2	15		
2	0.056	2	31		
2	0.067	2	20		
2	0.086	139	110		
2	0.082	140	110		
2	0.052	20	23		
3	0.082	2	3	15	
3	0.052	2	3	31	
3	0.082	139	140	110	
4	0.082	138	139	140	110

The association rules are then subject to correlation analysis, since rules with high support and confidence values may sometimes be misleading. In the proposed framework the lift correlation measure is calculated for each of the resulting association rules. Let an association rule $A \Rightarrow B$, the lift corresponds to $lift(A \Rightarrow B) = P(B|A)/P(B)$, or $lift(A \Rightarrow B) = confidence(A \Rightarrow B)/sup(B)$. The numerator is the likelihood of a host requesting both, while the denominator is what the likelihood would have been if the two visits were completely independent. Values greater than one indicate positive correlation between the two itemsets, values less than one indicate negative correlation and values equal to one indicate independence of A and B [7]. Correlation analysis will output $A \Rightarrow B$ [$support$, $confidence$. $Lift$] strong association rules to be used for the recommendation engine.

A sample of association rules is presented in Table 5, after the application of the measurements previously analyzed. The second row, for example, is interpreted as if a user requested Web page 140, he will also request Web page 139. This browsing behavior appears in 13 % of the data set transactions, i.e. user requests and the conditional probability that a user having requested 140 will also request 139, is 97 %. In addition, the lift value indicates positive correlation between the two itemsets as its value is greater than 1.

Table 5 Sample association rules

Antecedent		Consequent		Support	Confidence	Lift
2		3		0,280	0,940	3,350
140		139		0,130	0,970	7,660
140		139	138	0,130	0,970	7,660
3	20	2	15	0,070	0,084	8,370
138		110		0,090	0,840	8,470

Table 6 Sample association rules grouping

No	Antecedent				Consequent		
1			2	⇒	3		
2			15	⇒	2	15	23
3			15	⇒	2	20	
4			15	⇒	2	23	
5			15	⇒	2	110	
6			20	⇒	2	164	
7			20	⇒	2	110	164
8		2	15	⇒	3		
9		2	15	⇒	3	20	
10		2	15	⇒	3	20	23
11		2	15	⇒	3	23	
12	2	3	15	⇒	20		
13	2	3	15	⇒	20	23	

4 Prototype

In this section the preparation of the system prototype is presented. The association rules produced from the data mining process are used for the construction of the recommender system. The association rules whose lift is less than 1 or 1, are discarded as these values indicate negative correlation or independence of the itemsets, respectively. From those with *lift* value greater than 1, some will be selected based on their *support* or *confidence*. High *support* indicates that the rule appears often in the transactions data set, and high *confidence* indicates increased probability for the consequent of the rule to appear together with the antecedent. All association rules that satisfy the *support* and *confidence* thresholds that were set during the mining process, they can be used for the recommendation system, but depending on their values and their total number, the rules may be further filtered to reduce their multitude and keep the strongest ones. For example, if all the rules have high *confidence*, then they can be sorted based on their *support*, to choose the highest valued ones and discard the others. In this framework we selected the rules with high *confidence* since the aim is to provide accurate recommendations to the user. The set of association rules that will be finally used for the recommendation engine is then grouped based on the number of items in their antecedent. The result is a group of rules with one item in the antecedent, a group with two items and so on. Table 6 illustrates a sample of grouped association rules ready to be used for the construction of the recommender system. The one-rule groups are listed first, with their corresponding consequent, following the two-rule groups and so on.

The recommendation system captures the user's request, searches the one-item antecedent for a match and, if found, recommends the items in the consequent column. When two Web pages accessed by the user are known, it searches the one-item

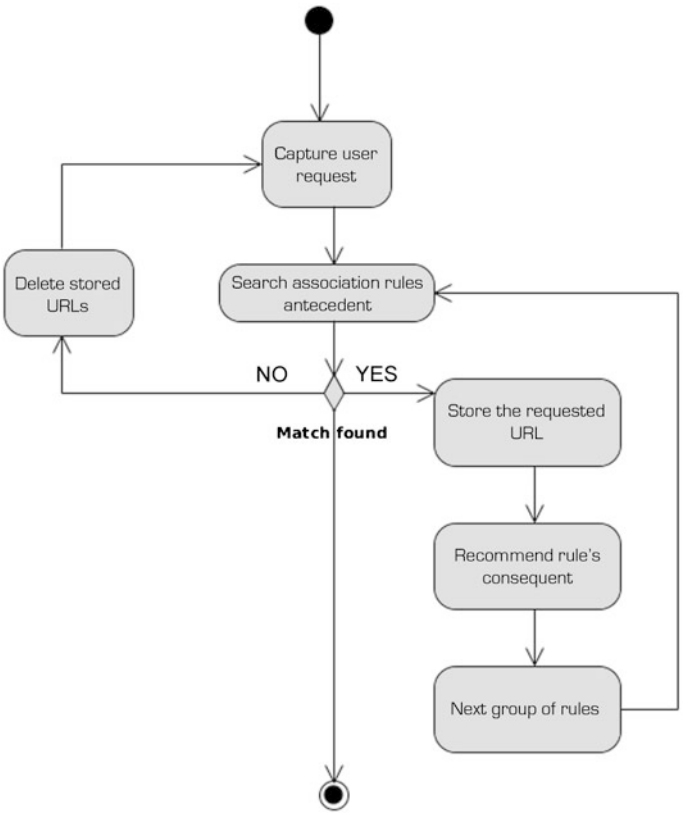


Fig. 4 Recommendation algorithm activity diagram

group for the second Web page and in continuance searches the two-items group for both consequent requests. When no match is found the system erases the user's browsing history and starts tracking it again when a match is found. The recommendation algorithm for the proposed framework is illustrated in the activity diagram of Fig. 4.

Following, the recommender system can be implemented either to run on a Web or proxy server, or on the client. In the first approach, a database server is also needed to store the user IP and the respective Web pages accessed as well as the association rules. In the second approach which was chosen for the system prototype, there is no need to track the different users, but the recommendation engine and the association rules need to be installed on each host.

The proposed framework was applied to real data collected through the log of a Microsoft ISA server used for Internet access from an organization of 250 employees.

The recommender system was implemented as an extension for the Google Chrome Web browser, using JavaScript and the Chrome API. The extension's ar-

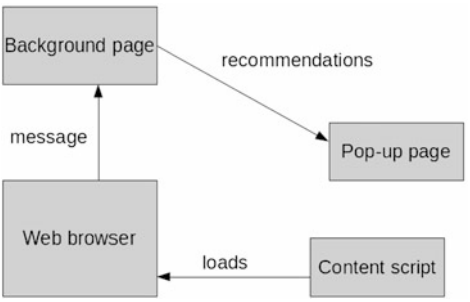


Fig. 5 System prototype architecture

chitecture is depicted in Fig. 5 and has four main components:

1. manifest.json: contains the necessary information for the installation of the extension.
2. contentscript.js: gets loaded when a new browser window or tab opens and gets executed each time a request is performed.
3. backgroundpage.html: runs in the background and implements the recommendation engine.
4. popup.html: pops up when the user clicks on the extension's icon and displays the recommendations.

The above components can be installed in a single directory and then a *browser action icon* is displayed. When a user opens the Web browser the *contentscript.js* is loaded and executed each time a URL is requested. It sends a message to the *background.html* page calling a Chrome API function. The *background.html* page has a listener that waits for messages from the content script. When the message is received the listener calls a JavaScript function that implements the recommendation algorithm. It then captures the requested URL and searches in the antecedent of the rule, and if the URL is found then it recommends the consequent of the association rule. This process runs in the background and when the user wants to present the recommendations, he/she just clicks on the *browser action icon* causing the *popup.html* to pop up and present the recommended URLs. At the top-right of Fig. 6, two recommended URLs are presented in the popup window and by choosing one of them the referencing Web page opens in a new tab. The *popup.html* is an HTML page, so it is easy to be extended or further formatted, adding CSS, JavaScript, etc.

The application runs only in the user's browser and monitors only the user's usage data, so there is no need to distinguish between different users as in most recommender systems. It also has the advantage that the user can explicitly enable or disable the extension, through his Web browser extensions manager and inspect the underlying code, assuming he/she wishes to. The recommender system captures only the current requests made by the user and tracks them, stores them, only while they are found to the association rules antecedent. In any other case they are deleted and no data are stored locally. The Web browser history also is not accessed at all. This approach is not expected to raise any privacy issues, as no code is executed in the foreground or the background, unless the user wishes to. In addition, although

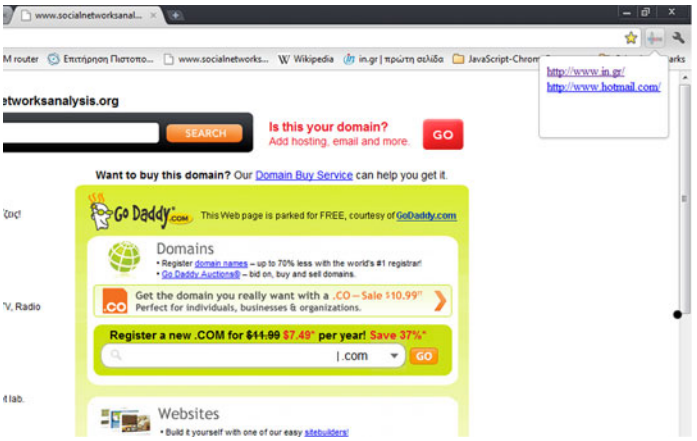


Fig. 6 Snapshot of the pop-up page

the engine generates recommendations, they are presented to the user only after the extensions icon, the *browser action icon*, is clicked.

The association rules are stored in a JavaScript file, rules.js, which can be easily replaced with an updated one. In this file, two JavaScript arrays are used to store the antecedent, *(left]* and consequent *(right[])* of the association rules. The recommendation engine searches the *left[]* array for a match and if found, recommends the corresponding value of the *right[]* array. The whole application is of small size, consumes the minimum resources and can be easily distributed and managed.

5 Discussion

In the proposed framework for developing a recommender system, social network analysis and data mining algorithms were applied on Web logs, aiming to improve the quality of Web page recommendations. The Web logs used, provided large amounts of Web usage data and their preprocessing was based on the techniques analyzed in [6], overcoming the problems related to proxy server and Web browsers caching. When the Web browser displayed a cached Web page, e.g., when the user probably revisited a page or tried to navigate back to a previous page by clicking the "Back" button, these pages were not considered relevant. Only the new requests made explicitly by the user were considered relevant, otherwise Web browser caching was ignored. In addition, a list of suffixes was used for the removal of the files that were implicitly requested by the browser and did not contribute to the format of the user browsing usage. Due to proxy server caching, requests possibly made by different users had the same identifier. The use of *cookies or user log-in* [6], were not applicable to the proposed framework, and since these records corresponded to a small portion of the data set, they were finally removed focusing on the dynamic explicit requests.

The next stage of the application of social network analysis algorithms revealed various peculiarities. Centrality measurements were efficient in identifying key nodes and their behavior, after the networks were exogenously affected. In combination with network level measurements they provided useful results on the networks cohesion and dynamics, benefiting the data mining process that followed. This phase was time demanding, as it included the construction of various social networks, of both users and Web pages, and their analysis in the time domain. The associations rules mining algorithms were applied to the data set, based on the results of the social network analysis, providing strong rules for the recommendation engine.

Related work on hybrid recommender systems, that merges social network analysis and data mining, constructs social networks of users linking them with the items of their interest, which are further analyzed to identify groups of people with similar behavior or interests. Friendship between users is assumed that indicates common interests, recommending items to a new user based on the items his/her friends have accessed [1, 8, 9, 13, 17]. The proposed framework differentiates from these approaches and introduces the novel approach of analyzing the social networks of users, in conjunction to the social networks of the Web pages. The key Web pages are identified and the way these networks are formed and evolve is analyzed, aiming to understand how they relate and affect each other.

In recommender systems up to now there is a common need to create user profiles, access the user's browsing history and store these personal data, in order to create recommendations. Privacy and security in this way is often affected, while the problem of effective user identification remains. In this paper, the resulting system prototype is a light-weight Web browser extension that implements the recommendation engine. Most popular Web browsers support extensions, so it could be easily ported to the majority of them. It alleviates privacy issues, since it is enabled explicitly on user demand and it does not search the Web browsers history but instead captures the Web page requests while enabled. Changing its architecture i.e., moving the recommendation engine on a server, could improve scalability and the performance of the system, as well as add new features as they are presented in the next section.

6 Conclusions and Future Work

A hybrid method for the construction of a Web page recommender system was presented that combines social network analysis and data mining to Web usage data which resulted in the development of a recommender system prototype. The process starts with the collection of the necessary data from the log files of the Web servers that host the requested Web pages, or the proxy servers which act as the intermediate between the users and the hosting servers or other data sources i.e., network traffic software that can capture and record the HTTP requests. This process could be performed either in the boundaries of an organization or in the Internet, having as output the corresponding log files.

The framework continues with the phase of data pre-preprocessing, which is critical for the quality of the data and consequently the results of the social network analysis and data mining algorithms. The exploitation of various data sources requires their integration, while data reduction and transformation are necessary due to the logs' format and the redundancy of data recorded in them.

The data set that results from the previous phase is in the form of multi-instance transactional data, where each transaction corresponds to a Web request made by a user either implicitly or explicitly. The specific dataset is used for the construction of the social networks of hosts and Web pages which are then analyzed calculating node and network level measurements. The analysis of the networks leads to the identification of the critical users and Web pages. This is achieved by separating the data to specific time periods and by analyzing and comparing various combinations of these data sets in the dimension of time. To measure importance, the analysis emphasizes on centrality measurements, and is combined with network level measurements, providing the complete description of the networks' structure and dynamics. The changes in the networks are detected using Statistical Process Monitoring (SPM) control charts e.g., CUSUM charts, and spectral analysis is applied to avoid erroneous conclusions caused by periodic events of the real world that affect and are reflected on the social networks.

Based on the results of the previous phase, the data set is divided and data mining algorithms are applied to mine for association rules. Data pre-processing transforms the multi-instance transactional data into single-instance, being the appropriate format for the data mining algorithms. FP-growth algorithm is then applied because of its performance on large data sets and the output is association rules which indicate patterns in the users browning behavior. The strongest rules are selected, based on their *support* and *confidence*, followed by correlation analysis to verify the strength of the rules. These association rules are further processed, by grouping them according to the number of items, i.e., Web pages, in their antecedent, and a system prototype uses them to create Web page recommendations for the users, to the prototype respects user's privacy, has limited non-functional requirements and its maintenance and extension is very easy. Future work may include the extension of the framework to include content-based filtering, explicit ratings from the users and classification of the users according to their usage behavior and preferences. The content-based filtering techniques can be used to correlate the Web pages already accessed by the user, with the ones recommended by the system, thus increase the quality of the recommendation. Additional functionality could be added to the system, enabling the user to explicitly rate a Web page, benefiting from the advantages of collaborative filtering. A weighted average [14] could be the result of the combination of the above techniques, enhancing also the quality of the recommendations. The Chrome extension could also be distributed, through the Google Web Store, gaining users' feedback from a human computer interaction perspective. The social networks constructed by the users and by the Web sites could also be further analyzed to identify how they influence and affect each other. A step further at the association rules preparation could be taken, clustering the rules and ranking them based on the confidence of the recommendation and correlating the groups of association rules. The

rules' clustering increases the search performance, which could be further examined in combination with the systems scalability. Software components could also be added to the system prototype, to support the features proposed above, in addition with changes in its architecture e.g., by placing the recommendation engine on a server, in order for the extension to remain light-weighted.

References

1. Baraglia R, Lucchese C, Orlando S, Serrano M, Silvestri F (2006) A privacy preserving web recommender system. In: Proc. of the 2006 ACM symposium on applied computing
2. Bonacich P (1987) Power and centrality: a family of measures. Am J Sociol 92:1170–1182
3. Borgatti SP (2003) The key player problem. In: Breiger R, Carley K, Pattison P (eds) Dynamic social network modeling and analysis. National Academy of Sciences Press
4. Borgatti SP (2006) Identifying sets of key players in a network. Comput Math Organ Theory 12(1):21–34
5. Carley KM, Reminga J, Storrick J, Columbus D. ORA user's guide CMU-ISR-10-120
6. Cooley R, Mobasher B, Srivastava J (1999) Data preparation for mining World Wide Web browsing patterns. Knowl Inf Syst 1:1
7. Han J, Kamber M (2007) Data mining: concepts and techniques, 3rd edn. Morgan Kaufmann, San Mateo
8. He J, Chu WW (2010) A social network-based recommender system (SNRS). Computer Science Department University of California, Los Angeles
9. Hsu WH, King AL, Paradesi MSR, Pydimarri T, Weninger T (2006) Collaborative and structural recommendation of friends using weblog-based social network analysis. AAAI Spring Symposium
10. Frantz TL (2008) Annual tools/computational approaches/methods conference, Carnegie Mellon University, March 19
11. Freeman LC (1979) Centrality in social networks I: conceptual clarification. Soc Netw 1:215–239
12. McCulloh I (2009) Detecting changes in a dynamic social network, March 31 CMU-ISR-09-104
13. Palau J, Montaner M, Lopez B, Lluis de la Rosa Jo (2004) Collaboration analysis in recommender systems using social networks. In: Proc. cooperative information agents VIII 8th international workshop, CIA 2004, Erfurt, Germany, 27–29 September
14. Vozalis E, Margaritis K (2003) Analysis of recommender systems' algorithms. In: Proc. of the 6th hellenic European conference on computer mathematics & its applications (HERCMA), Athens, Greece
15. Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, Cambridge
16. Xu G, Zhang Y, Li L (2010) Web mining and social networking: techniques and applications, 1st edn. Springer, New York
17. Zanin M, Cano P, Buldu JM, Celma O (2008) Information spread in recommendation systems. In: Proc. of the workshop Net-Works 2008, Pamplona, 9–11 June, pp 1–4

Evaluation and Development of Data Mining Tools for Social Network Analysis

Dhiraj Murthy, Alexander Gross, Alexander Takata, and Stephanie Bond

Abstract This chapter reviews existing data mining tools for scraping data from heterogeneous online social networks. It introduces not only the complexities of scraping data from these sources (which include diverse data forms), but also presents currently available tools including their strengths and weaknesses. The chapter introduces our solution to effectively mining online social networks through the development of VoyeurServer, a tool we designed which builds upon the open-source Web-Harvest framework. We have shared details of how VoyeurServer was developed and how it works so that data mining developers can develop their own customized data mining solutions built upon the Web-Harvest framework. We conclude the chapter with future directions of our data mining project so that developers can incorporate relevant features into their data mining applications.

Keywords Data mining · Online social networks · Web content extraction · Web-Harvest · Network analysis

1 Introduction

The practice of data mining and web-content extraction is an important and growing field. Many disciplines are looking at 'big data' and ways to mine and analyze this data as the key to solving everything from technical problems to better understanding social interactions. For example, large sets of tweets mined from Twitter have been analyzed to detect natural disasters [3, 8], predict the stock market [1],

D. Murthy (✉) · A. Gross · A. Takata · S. Bond
Social Network Innovation Lab, Bowdoin College, 7050 College Station, Brunswick, ME 04011, USA
e-mail: dmurthy@bowdoin.edu

A. Gross
e-mail: agross@bowdoin.edu

A. Takata
e-mail: atakata@bowdoin.edu

S. Bond
e-mail: sbond2@bowdoin.edu

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_10,

and track the time of our daily rituals [5]. As our use of blogs, social networks, and social media continues to increase, so does our creation of more web-based hyperlinked data. The successful extraction of this web-based data is of considerable research and commercial value.

Data mining often goes beyond information retrieval, towards a meta-discovery of structures and entities hidden in seas of data. As our social interactions become increasingly mediated by Internet-based technologies, the potential to use web-based data for understanding social structures and interactions will continue to increase.

Online social networks are defined as

> web-based services that allow individuals to (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system [2].

Individuals interact within online social networks through portals such as Facebook, which create social experiences for the user by creating a personalized environment and interaction space by combining knowledge of one users' online activity and relationships with information about other networked individuals. It is through data mining algorithms that Twitter, for example, determines recommendations for users to follow or topics which may be of potential interest. One way to study social networks is by examining relationships between users and the attributes of these relationships. However, data on a blog, Facebook, or Twitter is not inherently translatable into network-based data. This is where data mining becomes useful. Social networks typically only provide individual portal access to one's egocentric network. Put in the language of social network analysis (SNA), the visible network is constructed in relation to ego (the individual being studied) and relations of ego, known as 'alters', are seen (e.g. Facebook friends). However, in a restricted profile environment, the alters' relationships are not revealed. In order to understand network structure (which is key to a systems perspective), the researcher must use methods like data mining in order to gather information about all users and interactions by iterating over the data. A variety of different types of tools have been developed to collect this web-based information. These tools were created for a wide array of purposes. The majority of these tools have been commercially released. Some of these tools can be used to construct profiles of individuals based on data from multiple sources. Given issues of privacy, ethical uses of these tools should be strictly employed [15].

Despite the existence of a variety of tools, the simplicity and robustness of them varies widely. There are many types of networks and online communities that could qualify as a subject of network-based research. Many of these virtual organizations and networks often share key elements and structures that are common across online social networks. These could include users, groups, communications, and relationship networks between these entities. Also unlike the simple data that is subject of most data mining projects, SNA is not merely focused on generating lists of entities and information. Social networks are more organic in their growth and place emphasis on relational attributes. SNA seeks to understand how individuals and groups within networks (termed 'cliques') are connected together.

The Social Network Innovation Lab (SNIL) is an interdisciplinary research lab dedicated to understanding online social networks, social media, and cyberinfrastructure for virtual organizations. Research at the SNIL often involves the need for tools that are able to extract social network-based data for analysis from varied online social communities. The SNIL currently has projects which require data mining of popular microblogging services, shared interest forums and traditional social networks. We found that many currently available data mining tools were insufficient or poorly suited toward applications in social network research. This led us to begin to investigate the development of our own custom data mining tools. As part of this project, we researched existing tools, developed a conceptual framework for general data mining of online social networks, and tested prototype implementations of these ideas while acquiring data for use in current ongoing projects.

In this chapter, we will consider a variety of common methodologies and technologies for generic data mining and web content extraction. We will highlight a number of features and functionalities we see as key to effective data mining for social network analysis. We will then review several current data mining software tools and their fitness for data mining online social networks. The remainder of the chapter discusses our development of a data-mining framework for online social networks. Specifically, we introduce our work in extending the Web-Harvest 2.0 framework to data mine online social networks. This is followed by a case study of some of our initial results to acquire data from an online virtual community organized around social network technologies. The remaining sections summarize what we have learned through this process and maps out a course of action for future developments in this area.

2 Web-Content Extraction Technologies

In online social networking sites, the information and data that constitutes the network and its entities are, by necessity, distributed over a vast array of unique and dynamically generated page instances. Even when only a basic set of common SNS features (user profiles, friend lists, discussion boards) are considered, it is easy to see how the number of pages required to encounter and capture all the activity of the social network could quickly grow exponentially. In order to study the structure and operation of virtual communities within social networks, researchers need to parse and capture this sea of distributed data into formats more appropriate for research and analysis. In the absence of direct access to the database systems that drive social networks or a site-provided API, one must utilize other means to capture SNS data for research.

The majority of information on the Internet is circulated in the form of HTML content, which wraps information in a nested set of tags that specify how data needs to be visually rendered in the browser. This is suitable for making data easily read and understood from the screen or through printing, but not as useful when clean organized machine-readable datasets are desired. Most online data extraction tools

take advantage of the fact that the HTML is itself a structured data interchange format and leverage the HTML format to create parsers, which can extract the simple content of the page in an organized way while discarding the irrelevant material. Generally, most online data extraction technologies can be classified into several categories.

2.1 Formats, Conventions, Utilities, and Languages

Technologies in this class are low-level constructs that often derive from some sort of published standard grammar. This grammar may then be implemented in whole or in part by other higher-level technologies. They often simply define a way in which data can be ordered, searched, manipulated, or transformed. For instance, the XPath standard defines a format for finding and isolating pieces of information from a structured XML document. Similarly, regular expressions are a structured format, which are useful for performing advanced searches and manipulation on unstructured strings of characters. XSLT is a language defined to assist in the transformation of one type of structured XML document into another (e.g. transforming an HTML document into a simpler RSS feed or vice-versa). Without well-defined standards for interacting with various types of data, extraction becomes more difficult. However, because of the low level nature of these structures, they present challenges in isolation to when used in advanced extraction projects (without constructing a broader framework for their application to a set of data).

2.2 Libraries

Data extraction libraries often perform the job of wrapping one or more lower level data manipulation/extraction constructs into an organized framework within the context of a specific programming language. These libraries then manage the implementation of a given construct within a framework useful for further development within a programming language. Development libraries leave the end goals completely open to the developer. Depending on the time, investment, and goals of the developer, development libraries can be used to create anything from simple one-off scripts to high-level applications with many advanced features.

2.3 Web-Based APIs and CLUI

Web-based API and command line user interfaces (CLUI) provide a standardized abstraction layer to certain sets of web content. These typically wrap development libraries (with their exact nature dependent on the hosting server and application). Furthermore, they will generally apply and be structured around the content available from one data source (e.g. a particular website, web-enabled technology, or ap-

plication). A prominent example is Google's OpenSocial API framework [7]. The OpenSocial API is an open standard for a set of API features specific to social networking. Code developed against such frameworks would be potentially adaptable to any social network using the open standard. Other examples include closed APIs provided by popular social networking sites including Twitter and Facebook.

2.4 Applications

The vast majority of data extraction solutions take the form of applications. Applications make use of a large set of extraction technologies and development libraries and bundle them into an interface designed around a set of desired functionality. Depending on that set of functionality and the level of expertise expected of the user by the developer, there can be a wide range of different types of data extraction applications. These applications range from self-adapting, learning, fully GUI based extractors for non-technical users to applications for advance data extraction that require some in-depth knowledge of programming or data extraction utilities. Many applications fall into this latter category. Some of the most common are Helium Scraper, Djuggler, Newprosoft, Deixto, and Web Harvest.

2.5 Enterprise Suites

This class of data extraction solutions is characterized by providing very high level, multi-featured, and advanced software solutions. They are often delivered as a suite of highly specialized applications. The implementations of these software packages are usually not open-source (as the code is often developed from proprietary development libraries). Like many enterprise solutions, these software products are often intricately complex and necessitate special training and/or ongoing technical support from the company itself to effectively use these tools. This support and training usually is an additional cost beyond the original software license. Though costly, this support may allow the client to obtain custom solutions to their specific needs which would be developed for them by corporate developers in response to exact client needs. Pentaho (http://www.pentaho.com) and QL2 (http://www.ql2.com) are two examples of enterprise level data extraction and mining solutions.

2.6 Outsourcing, Contracting, and Crowdsourcing

Alternative paradigms that merit mention are outsourcing, crowdsourcing, and freelance contracting. In these models, researchers with data needs simply ask one or many people to help, (usually in exchange for a one-time fee based on delivery of

the desired data sets). In outsourcing and freelance work, the researcher partners with a company or individual and explains the data extraction project and agrees upon a price and timeline for the delivery of the data. Crowdsourcing uses a slightly different model where a large data mining task might be divided into a large number of small tasks and a small fee may be offered for the delivery of each incremental piece of data delivered. Amazon's Mechanical Turk is a popular platform for streamlining this process and has been successfully deployed [13]. Individual micro-tasks are constructed such as asking someone to record the tags on an online article or to classify a given webpage. Mechanical Turkers are often offered between 0.01¢ and 0.20¢ for the completion of each microtask. For some researchers, outsourcing works well because the tasks are cost-effectively completed in a short timeframe [13]. For researchers who do not regularly need to acquire new data, this one-time fee structure may work well. Crowdsourcing may also be cost- and resource-effective. This method can bring additional concerns of uncertainty of when the task will be completed and, more importantly, quality control of data as contributors usually very highly in terms of quality of work [14]. These approaches do not represent new technologies for data mining per se, but to illustrate new solutions for researchers in the absence of tools and expertise for acquiring their own data sets.

3 Considerations for Data Mining of Online Social Networks

The utility of data mining applications for social network research is dependent on what functionalities are most appropriate to the domain. This section explores what functionalities are most valuable in social network research. This discussion is guided and informed by experiences and needs identified through our own research in the study of life science virtual communities of practice as well as work exploring health related communication networks on Twitter.

3.1 Input and Output

Data extraction is ultimately about acquiring formatted information from a data source and then translating, manipulating or filtering this information into other formats as appropriate to one's research objectives. In basic online content extraction, the initial input is often a simple web address of the location of the information one seeks to capture. The distributed organization of most social networks means the information you need could be dispersed over a large number of pages. The URLs for these pages may need to be dynamically generated from one or more lists of attributes. Furthermore, one will most likely need to extract information from one location and use the results to identify and locate other pieces of information. This paradigm is best served by tools which allow for the most possible types of automated data extraction and manipulations. Data mining tools should be able to

both read from and output to as many potential data formats as possible. Common formats most useful within a data mining tool kit include various kinds of structured text files like HTML/XML, delimited text, spreadsheets, or JSON. Data mining tools can be even more powerful when they include the ability to read and write to database systems, APIs, or even to have the ability execute local system commands to generate input variables. The power and usability of a tool increases as it is able to take input and give output in diverse formats.

3.2 Dynamic Query Specification

An important feature to consider when evaluating the utility of a data extraction tool to one's research needs is to understand the ways in which the tool allows you to request and gather information. Many basic data mining applications use a GUI to allow one to specify the desired extractions. This will always have certain limitations. Other tools use a command-based query language like SQL to scrape data. In our work with social networks we often need to traverse a list of forum locations, record all the user names encountered, collect information from each user's unique profile page, and then conditionally acquire extra information about certain users (if they are found in the previous step to have some specific attribute). Queries of this complexity often cannot be defined as a single request without resorting to multiple individual queries managed by a researcher. A query of this sort requires grammar for conditional branching, looping structures, and benefits from the ability to define functions (as well as local and global variables). Tools that implement full programming logic allow complex, dynamic, and context-aware queries to be defined. All the entities from a social network could conceivably be acquired via one request. This frees the researcher from having to micromanage many aspects of a complex data mining project.

3.3 Social Network Interfacing

Online social networks often recognize the importance of allowing access to their data. Site developers often provide a programming interface for third parties developers to create value-added applications leveraging this social data (the use of which might further enhance participation in the network by its users). These application programming interfaces (APIs) often provide alternate methods for requesting information from a site beyond simply observing the information in situ. For example, Twitter and Facebook both have well-developed APIs to access vast stores of data. As opposed to simply requesting a page and extracting data from it, APIs allow developers to make a special kind of request to the API and return just the raw data one is looking for. APIs can greatly ease the process of gaining access the information on a social network. Some of the smaller, less well-known, networks we are studying include API-like features. Data mining tools for the analysis of social networks benefit greatly if they allow content extraction from common APIs.

3.4 Job Scheduling

In contrast to many common data mining tasks, social network researchers are more likely to be interested in near real-time data. Using the methods already described, one could create a data mining specification which would capture a snapshot of the activity on a social network at the time the extraction was run. But, what about the next day or a month later? The constant use and modification of networks by their user base can cause networks to change greatly in short periods of time. One could simply capture a snapshot once a month, but these snapshots could contain large amounts of redundant information. A more robust solution might involve using programming logic coupled with automated time aware functionality to develop a data extraction request with the power to detect changes within the source network and incrementally update itself over time. In addition, a system of this type would likely require a perpetual extraction process which is able to detect and log changes periodically, and then call appropriate update scripts autonomously as needed. There are several key types of job scheduling attributes which could be associated with data extraction scripts:

Now — This option would immediately execute a job. This is the most basic type of scheduling operation.

Later — This option allows a user to schedule jobs at specific times. This could be useful to extract data from a site during low traffic hours, or for a situation where it is known that new information will be posted or made available at a specified times.

Chain — The ability to chain tasks would allow one job to be scheduled to start once another has completed. This is very useful when one data extraction task is dependent upon the completion of one or more other tasks. With this option, the whole extraction flow for a complex and self updating extraction job could be specified in advance and sent to the mining application as a single project.

Recurring — Recurring jobs are valuable in data mining of online social networks. Social network data presents challenges due to the fact that social networks are often in continuous flux. Most research of online social networks begins by capturing a snapshot of data as it exists at a specified point in time. A robust function for recurring data extraction allows researchers greater control over when to update their data and at what intervals to poll the site for new information.

3.5 Concurrency

Most web-content extraction tools acquire data by creating a virtual agent to make automated requests from a web host. This is the same way a web browser works. A browser requests a web site and the host sends a file containing information (i.e. in

HTML) needed to display the web page. In data mining, this data is simply grabbed and parsed in a variety of methods to obtain data. Most tools and extraction workflow simply utilize one agent to fetch each required page in sequence. For non-dependent extraction tasks, this process could be streamlined by creating multiple agents to make multiple concurrent requests to the same web host (for different information). For large jobs, this feature places a key role in speeding up the acquisition of data, (as these requests are rarely taxing on local hardware or network speed). Concurrency should be considered a key component of any data mining tool designed for online social networks.

3.6 Progress Management

Many of the features we have discussed focus on ways to allow for some level of automation in a data extraction task to be specified in advance (so the researcher is not required to micromanage the numerous aspects of large and complex extraction jobs). Usually, the analysis of social networks requires collecting large amounts of data from a network. In data mining tasks, data extraction is often limited by the speed that the hosting server allows clients to access data. Aside from concurrency, there is often little that can be done if the social network you are mining is slow at returning requested information or is very large. If the network is both, it could take hours or perhaps days to complete certain data extraction tasks. Our research often involved extracting information from numerous user profiles and forum pages. We noticed that although we could not predict when a job would complete, we could often determine the number of entities that needed to be captured before the process took place. This helped us realize that if we could also keep track of the number of completed entity extractions, we would be able to implement a progress monitor and estimate completion time for all our extraction tasks. We feel this type of progress tracking and management are important features for the data mining of online social networks. Wherever possible, extraction tools should attempt to keep track of the progress of data extraction tasks as well as expected time to completion. This feature will be of great value to researchers responsible for managing one or more large data extraction projects by giving them the information they need to maximize their own productivity and to be prepared in advance as to when data will be ready for post-hoc processing.

3.7 Extraction Meta-behavior

A natural instinct in the design of data extraction tools for large social networks is to find ways to acquire the desired data as quickly as possible. Further consideration reveals an important counter argument to this instinct. The operator of an extraction process tool might be tempted to create large numbers of page request agents,

which in turn might generate large amounts of traffic on the hosting site. This not only is considered bad 'netiquette' [11], but has ethical and legal considerations. If one's data mining project is part of academic research, the relevant Institutional Review Board (IRB) should be consulted to confirm ethical compliance with human subjects. A large volume of page requests with a web host could degrade the quality of the experience of other users of the site. Also it could result in the web host banning all requests from your IP address if the host believes your requests are malicious. Furthermore, many social networking sites have specific policies or Terms of Service (TOS) in place that would dictate how much data can be requested per agent. Whenever possible, it is recommend that permission and guidelines be obtained from the administrators of any site one wishes to extract information from. It is in the best interest of the data extractor to be responsible and follow any appropriate rate-limiting conventions whether explicit, or implicit when extracting data from a host. Having one's IP banned from a site could be potentially catastrophic to a research project. Any data mining toolkit for online social networks should implement some standard to protect the user, but also provide the ability to create custom guidelines depending on the known TOS of a web host or for when the user knows it is acceptable to request large volumes of data. These limits should be able to be defined in multiple ways by the number of requests per time unit or staying below some defined throughput or bandwidth measure. The idea is to be able to maximize program efficiency to acquire data as fast as possible, but not so fast that you may be garnering ill-will from a network and its users. In other words, data mining social networks should be—if possible—an open and collaborative process. Often, the social network being studied will also be interested in research results gained from the data which was mined.

3.8 Client-Server Paradigm

Extracting data from the web can require significant processing power as well as bandwidth. Many types of data extraction projects may be ongoing and most users do not want their computer constantly running potentially resource expensive scripts. We feel an ideal solution involves tools that use a client server paradigm (where each user simply submits their jobs to a server for handling). That way, the designated server can handle all the heavy processing and high data load while the clients' machine remains free for use. With robust scheduling, concurrency, and progress management in place, the server application just needs to notify the client when the final collected data is available. The use of a client side application gives a lot of flexibility to the user requesting certain extraction jobs. They can use the client to log onto the main server and manage all their running jobs regardless of where they are physically located. The server should provide the client with options such as checking job progress, creating new jobs, aborting running jobs, changing scheduling, and changing the extraction specification. Also, this provides the ability for multiple users with different data extraction needs to utilize one centralized server.

4 Review of Existing Data-Mining Tools to Mine Online Social Networks

Our ultimate goal in this research was to begin development of a custom data mining solution optimized for extracting information from online social networks. Though we began the process with the aim of developing a custom built-from-scratch data extraction toolkit, we decided to first evaluate existing tools. As discussed in the previous section, we were interested in whether we could incorporate these existing tools into a hybrid data extraction toolkit which implemented our of functionalities. In this section, we present this evaluation of common online data extraction tools. These reviews speak both to a tool's usefulness in social network research and in regard to their ability to be used as the basis for a prototype extraction tool. After the evaluation of several commonly available tools and technologies for online data extraction, we determined that Web-Harvest 2.0 was an ideal choice for the needs of our project goals. Among the tools considered were Helium Scraper, Newprosoft, Happy Harvester, Djuggler, Rapid Miner, Deixto, and Web-Harvest.

4.1 Common Data-Mining Tools

Based on our evaluation it was determined that Helium Scraper, Newprosoft, Happy Harvester, and Djuggler were all powerful GUI-based scraping applications. However, these tools also shared the same limitations. All four tools were single operating system applications that only allow scraper configurations to be defined within the context of the application. They also have no ability to be controlled or configured from the command-line. Their source code is not open-source and scripts could not be written against their various executables. When taken in consideration with our project goals (which would require software modifications for large social network scrapes to be conducted with minimal impact on the host), it became clear that these tools could not be leveraged to achieve our desired functionality.

4.2 Rapid Miner

Rapid Miner is one of the leading open-source applications for data mining and analytics and has been successfully used in data extraction projects [4, 6]. Rapid Miner was evaluated as a potential fit for our project's needs. It is open-source, cross-platform, uses XML-based configuration files, which can be developed through the interface or written directly, and the code base can be scripted against both in application wrapper interfaces as well as from the command line. Though Rapid Miner is a powerful tool, it has a steep learning curve and includes a large number of features which would not be needed for our project's needs. Using Rapid Miner would prevent us from being able to develop a fast lightweight utility in a reasonable amount of time.

4.3 DEiXTo

DEiXTo (also known as ΔEiXTo) is another web extraction technology that was evaluated for our project's needs [10]. DEiXTo is a single platform GUI-based web extraction application built on top of an open source Perl-based scraper utility. DEiXTo also uses an XML based configuration language that potentially allows configurations to be defined outside of the GUI. The Perl module that forms the backbone of DEiXTo's extraction technology could also be scripted against on any operating system or code framework which supports the Perl scripting language. The DEiXTo file format (.wpf) is obtuse and the documentation is not particularly accessible. This means that most .wpf files must be developed within the GUI application, which is single platform and is not open-source. DEiXTo is also limited in the ways output can be written only to specific file formats and in specific ways. While the features and options available in DEiXTo would allow us to accomplish our project goals, it was determined that tools which were more configurable and more open (in terms of input and output capabilities) would be better suited to our project.

4.4 Web-Harvest

Web-Harvest 2.0 is a Java-based open source data extraction tool, which has been successfully used in the data mining literature [16]. It is a hybrid tool that consists of a GUI based application wrapped around an open-source Java development library. This library, in turn, implements several of the most common and powerful extraction utility formats such as XPath and regular expressions. The Web-Harvest 2.0 platform also defines syntax for defining custom data extraction workflows. This was ideal for several reasons. First, quick start-up of development was possible using the features of the graphical user interface to easily debug, learn, and understand how to develop complex workflows in the Web-Harvest scraper configuration format. In many ways, defining workflows via this format is better than coding library solutions that require workflows to be defined in the context of that code base. This is because the configuration syntax is just a simple standard which can be written with any text editor. This frees the developer from the additional nuances of any high-level programming language. Furthermore, once tested, these workflows could be easily shared with others and passed to the development package, which could execute the scraper configurations through code. The fact that at the core of Web-Harvest is an open-source data extraction engine, allowed for our project to wrap this engine in our own lightweight code. Web-Harvest 2.0 was a good fit because of its hybrid nature. Most pure application based scrapers are not extendable, and few define a configuration format (forcing the developer to work within the confines of what the application allows). Pure development package based extraction tools can have a steep learning curve and are often difficult to debug. Relying purely on data extraction utilities and standards like XPath and regular expressions requires that an

entire framework be built around them in order to execute complex dynamic extraction workflows. This can be time-consuming and resource intensive. Given the remit of our project, Web-Harvest served as an ideal solution in terms of features and the ability to separate between code and UI (which allowed us to quickly develop our own tools using the Web-Harvest engine). Other tools were not found to efficiently allow us to work in this way.

5 Extension of Web-Harvest for Data Mining of Online Social Networks

After a review of existing data mining tools and a consideration of the desired features of data mining for online social networks, we decided to develop extensions and an application wrapper for the open-source Web-Harvest 2.0 data extraction engine to serve as a prototype of a full-fledged social network-centric data extraction engine. As discussed previously, Web-Harvest features a robust query specification language with capabilities to import and export data to a number of important formats including MySQL database integration. We sought to add features important to data extraction for social networks including time-based and repeating jobs, running multiple jobs simultaneously, client-based process and progress management, and server-based execution. Our tool adds all of these features by building on top of the open-source Web-Harvest source code.

5.1 Related Work

As discussed in the previous section, Web-Harvest has been used successfully in various studies as a basic scraper. One example is Nagel and Duval's [12] work, which used Web-Harvest in order to collect large amounts of publication data. For their study, they required a simple web scraper and used Web-Harvest in its original form to mine publication data from Springer, an academic publisher. They used the software to collect data including titles, authors, affiliations, and postal addresses.

Katzdobler and Filho use Web-Harvest extensively [9]. They combined Web-Harvest with JENA, a tool used to build semantic web applications, as well as an ontology which described what type of information they wanted to extract. The JENA API then accesses the ontology and Web-Harvest extracts the information from the site. However, manual creation of the configuration file and manual startup of Web-Harvest is needed.

TagCrawler is a program written using Web-Harvest and is one of the few cases of Web-Harvest being directly extended [17]. The creators of TagCrawler required a web crawling tool which would be able to retrieve information from tagging communities. While the end goal of the project was not related to our project, their use of Web-Harvest as a base and building from it illustrates that this can be a successful model.

5.2 Voyeur Server Project

Our project, 'VoyeurServer', uses Web-Harvest's existing functionality, but adds several layers of additional features. The features we identified in Sect. 3 served as a guide to our development efforts. By extending the Web-Harvest core framework, we were afforded the opportunity to discard the overhead of a GUI in favor of a lightweight command-line interface implemented with a client-server pattern. We were also freed from the limitations placed on the extraction engine by the GUI. We were also able to take advantage of the Java programming language to develop our own features not present in the GUI or the engine itself.

5.2.1 Structure

The implementation of this project using the client-server paradigm required that we split the functionality for extraction into two applications: the VoyeurServer and the VoyeurClient. The latter was developed as a lightweight command line interface to provide users a way to submit and control their individual data extraction projects. It is through the VoyeurClient that users can submit their extraction jobs, manage the job's behavior, as well as monitor the progress of all their running jobs. The VoyeurServer application was designed to run continuously on a special server and to respond to requests from VoyeurClient instances. When the server receives a job from a client, the server creates an instance of the Web-Harvest 2.0 extraction engine in its own thread, and manages the execution of the job as directed by users' interaction with the client. Figure 1 shows the relationships and flow of communication between these program entities.

5.2.2 Functionality

In addition to the ability to submit and manage multiple simultaneous extractions, we also sought to develop a set to features to allow for the smart management of extraction behavior and management of job progress (important functions outlined in Sect. 3). The ability to control and manage this behavior was incorporated into the VoyeurClient and VoyeurServer applications.

5.2.3 Temporal Control

The ability to control the temporal execution of job and recurrent behavior is one of the key features we identified for social network-aware data-extraction. This control allows users to create self-updating scripts as well as control how and when individual jobs will be run. Taking a cue from modern calendar systems, VoyeurServer was designed to allow jobs to defined as a single event or as a repeating process to be executed in an ongoing manner.

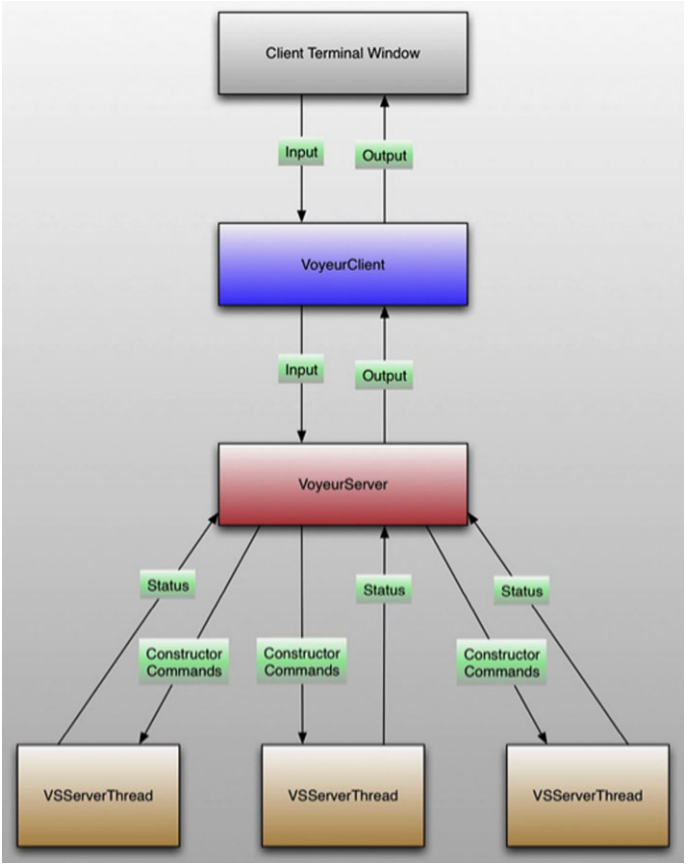


Fig. 1 Schematic of data flow in VoyeurServer

5.2.4 Process Management

VoyeurServer users retain full control over their running and completed jobs. There is some danger that clients might feel they have lost control once their jobs were submitted to the server. In order to combat this, the VoyeurClient provides users with complete control over their running jobs at all times. This functionality includes the ability of users to start, stop, or pause as job at any point during its execution. Additionally users are given the ability to alter or remove a jobs repeating behavior at any time.

5.2.5 Progress Management

The ability to view and manage the progress of ongoing scrapes is an important feature for users managing large and complex data extraction jobs within online social

networks. In order to add this functionality to the existing Web-Harvest extraction framework, we developed an extraction specification template for progress aware content extraction jobs. This template adds two requirements to any extraction file. The first requirement is the initial definition of some measure of progress, which is stored in a special progress management database. The second involves having the extraction specification itself update this database regularly (based on its ongoing progress). The addition of these structures to any Web-Harvest extraction specification file is sufficient to make the job progress-aware within the VoyeurServer framework. At the user's request, the VoyeurClient application will request and report this information to the user about their progress-aware jobs. Additionally, this feature will report on the start time, current running time, and any important log messages of any running job (regardless of whether it implements the progress-aware framework).

5.3 Project Summary

There already exist valuable tools which can be modified for many data mining projects. In our case, we found that Web-Harvest 2.0 already incorporates many of the basic functionalities identified as important in mining online social networks. Because the code is open source, we saw Web-Harvest as an ideal place to begin testing and developing a social network-centric data mining tool. Our development plan centered on taking the core framework of Web-Harvest 2.0 and wrapping the code base to extend the application to serve as a multithreaded data extraction engine (implemented using a client-server interaction paradigm). Once the base extraction modules were wrapped in this way, we could focus on adding additional management features to the wrapper like task scheduling and process/progress management. Further work to develop and refine these ideas will be important to make this tool available to the broader data mining community.

6 Experimental Results: A Case Study

We engaged in this exploration of developing data extraction tools better suited towards collecting data for online social networks as part of a NSF funded research project on Virtual Organization Breeding Environments (NSF grant #1025428). A key aspect of this project required us to gather large amounts of data from online life science communities of practice in order to explore the organization and cyberinfrastructure of virtual communities within these networks. As we collected this data, we realized there were ways in which the functionality of existing data extraction tools could be enhanced to streamline our extraction workflows.

The online network we studied had several key features we were interested in capturing. Specifically, we needed to collect over 200,000 user profiles, over 9,000

forums, a blogging ecosystem, and a friendship-based social network. This section details how we were able to use our tool to acquire the information we needed for our research.

Our needs included capturing information about the users within this community and their communications with one another. Our eventual goal is to use this data to study patterns of trust development and of online scientific collaboration. The community we studied did have an API for requesting information about each user's egocentric peer network. However, for all other data we had to rely on traditional web content extraction methodologies.

We were able to use the Web-Harvest query specification language to develop separate workflows to collect each type of information (user data including profile information, forum posts, friend networks). Using the features developed in our VoyeurServer application, we were able to develop a broad automated workflow to simultaneously collect data from these three key areas (while at the same time respecting the bandwidth consumption limits agreed upon by communication with site administrators). Using the workflows and the VoyeurServer tool, we were able to successfully collect user and post information over the course of a week while limiting any daily micromanagement of the process. In this regard, our time aware processes, progress management features, and concurrent extraction jobs were able to prove themselves successful in application. The flexibility of Web-Harvest's I/O framework allowed us to capture data to structured XML as well as directly to a database simultaneously. In our research, we had previously developed an application to assist in the qualitative coding and classification of the community's data in this database. Our VoyeurServer content extraction workflows allowed the data to be integrated directly with our existing downstream research applications. This realized potential represents an extremely powerful and desirable workflow for network analysis. This experiment also helped identify issues that were not addressed by the current version of VoyeurServer. Some of these limitations and potential solutions are discussed in the next section.

7 Future Work

Despite initial success in using Voyeur Server for mining data from online social networks, there remain further capabilities of Voyeur Server that require further development. In Sect. 3, we outlined key features for data mining of online social networks. Currently, the VoyeurServer extension of Web-Harvest implements these features at a basic level. Further testing and development would help determine whether this extension has a future as a general research tool or whether it suggests that extending Web-Harvest 2.0 is perhaps less preferable than starting from scratch to develop a data mining toolkit for online social networks. We consider further work in robust data-extraction tools for social networks to be of the utmost importance. We are willing to share more details as well as our code upon request to help further these goals.

For those considering developing their own custom wrappers for the Web-Harvest 2.0 extraction engine, we suggest considering these suggestions of extended functionality:

- Utilize Web-Harvest's plugin architecture to develop integrated modules for common social network APIs to ease the process of developing extraction specifications for complex APIs (especially those requiring authenticated requests).
- Develop custom plugins for common database tasks. VoyeurServer and Web-Harvest database features rely on raw SQL statements and thus increases the level of expertise required to develop and implement database functionality.
- Develop a specification file for projects as opposed to per file scrapes. Incorporate time-aware functions, extraction meta-behavior, and progress monitoring options into this project specification format. This would allow for integrated individual workflows for large projects.
- Explore automated concurrency as opposed to having to design your individual jobs or projects for concurrency. This would help basic users take maximum advantage of parallel processing possibilities.

Our continued research seeks to address some of the issues and limitations discovered in developing this tool. We seek to further develop VoyeurServer in the following ways:

- Develop further functionality that allows for higher levels of concurrency.
- Investigate the feasibility of a broad-based public research server providing network-structured extraction as a service.
- Investigate high per-thread resources. Experience suggests that VoyeurServer is memory intensive. Our goal would be to reduce the memory overhead to a minimum for each running job. This will make running large numbers of jobs for various projects more efficient.
- Improve the interface for this Web-Harvest extension. Specifically, develop a GUI for the VoyeurServer client application

We have shared these improvements so that data mining developers can be aware of issues we currently face and some possible solutions. This will enable designers and developers to learn from the development challenges we have faced.

8 Conclusion

This chapter has reviewed various data mining tools for scraping data from online social networks. It has highlighted not only the complexities of scraping data from these sources (which include diverse data forms), but also introduces currently available tools and the ways in which we have sought to overcome these limitations through extensions to existing software. After reviewing data scraping tools currently on the market, we developed a tool of our own, VoyeurServer, which builds upon the Web-Harvest framework. In this chapter, we outlined the challenges

we faced and our solutions. We also included future directions of our data mining project. Concrete methods for developers to develop data mining solutions of online social networks using the Web-Harvest framework are provided.

Although this research is preliminary and its remit has not been to test all the features we have identified as being important to data mining online social networks, our experience in developing the VoyeurServer tools has been positive and represents what we believe to be an important step towards the further development of this and/or other data mining tools specifically for online social networks. It is important to begin developing these domain specific solutions so that good open source options are available to researchers. Current tools tend to be focused around the domains of marketing and business knowledge. These types of solutions will usually fall short for use in academic research. As social networks continue to become increasingly part of our online interactions, methods for data extraction from these networks will continue to remain important.

References

1. Bollen J (2011) Twitter mood as a stock market predictor. Computer 44(10):91–94
2. Boyd DM, Ellison NB (2008) Social network sites: definition, history, and scholarship. J Comput-Mediat Commun 13(1):210–230
3. Doan S, Vo B-KH, Collier N (2011) An analysis of twitter messages in the 2011 Tohoku earthquake. arXiv:1109.1618v1 [cs.SI]
4. Graczyk M et al (2009) Comparative analysis of premises valuation models using KEEL, RapidMiner, and WEKA computational collective intelligence. In: Semantic web, social networks and multiagent systems. Springer, Berlin, pp 800–812
5. Golder SA, Macy MW (2011) Diurnal and seasonal mood vary with work, sleep, and daylength across diverse cultures. Science 333(6051):1878–1881
6. Han J, Rodriguez JC, Beheshti M (2008) Diabetes data analysis and prediction model discovery using RapidMiner. In: Proceedings of the 2008 second international conference on future generation communication and networking, vol 03. IEEE Comput Soc, Los Alamitos
7. Häsel M (2011) OpenSocial: an enabler for social applications on the web. Commun ACM 54(1):139–144
8. Hughes AL, Palen L, Sutton J, Liu SB, Vieweg S (2008) "Site-seeing" in disaster: an examination of on-line social convergence. In: 5th international ISCRAM conference
9. Katzdobler F-J, Filho HPB (1999) Knowledge extraction from web. In: Book knowledge extraction from Web. http://subversion.assembla.com/svn/iskm/FinalDocumentation/FinalReport.pdf. Cited 15 Jan 1999
10. Kokkoras F et al (2008) MOpiS: a multiple opinion summarizer artificial intelligence: theories, models and applications. Springer, Berlin, pp 110–122
11. Morzy M (2011) Internet forums: what knowledge can be mined from online discussions. In: Knowledge discovery practices and emerging applications of data mining: trends and new domains. IGI Global, Hershey, pp 315–336
12. Nagel T, Duval E (2010) Muse: visualizing the origins and connections of institutions based on co-authorship of publications. In: Proceeding of 2nd international workshop on research 20 at the 5th European conference on technology enhanced learning sustaining TEL, pp 48–52
13. Sheng VS, Provost F, Ipeirotis PG (2008) Get another label? improving data quality and data mining using multiple, noisy labelers. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, Las Vegas

14. Snow R et al (2008) Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In: Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, Honolulu
15. van Wel L, Rotakkers L (2004) Ethical issues in web data mining. Ethics Inf Technol 6(2):129–140
16. Yang L et al (2009) Discovering topics from dark websites. In: Computational intelligence in cyber security. CICS '09. IEEE symposium. IEEE: Univ. of Tennessee at Chattanooga, Chattanooga, pp 175–179
17. Yin RM (2007) TagCrawler: a web crawler focused on data extraction from collaborative tagging communities. University of British Columbia, Vancouver

Learning to Detect Vandalism in Social Content Systems: A Study on Wikipedia

Vandalism Detection in Wikipedia

Sara Javanmardi, David W. McDonald, Rich Caruana, Sholeh Forouzan, and Cristina V. Lopes

Abstract A challenge facing user generated content systems is vandalism, i.e. edits that damage content quality. The high visibility and easy access to social networks makes them popular targets for vandals. Detecting and removing vandalism is critical for these user generated content systems. Because vandalism can take many forms, there are many different kinds of features that are potentially useful for detecting it. The complex nature of vandalism, and the large number of potential features, make vandalism detection difficult and time consuming for human editors. Machine learning techniques hold promise for developing accurate, tunable, and maintainable models that can be incorporated into vandalism detection tools. We describe a method for training classifiers for vandalism detection that yields classifiers that are more accurate on the PAN 2010 corpus than others previously developed. Because of the high turnaround in social network systems, it is important for vandalism detection tools to run in real-time. To this aim, we use feature selection to find the minimal set of features consistent with high accuracy. In addition, because some features are more costly to compute than others, we use cost-sensitive feature selection to reduce the total computational cost of executing our models. In addition to the features previously used for spam detection, we introduce new features based on user action histories. The user history features contribute significantly to classifier performance. The approach we use is general and can easily be applied to other user generated content systems.

S. Javanmardi (✉)
University of California, Irvine Donald Bren Hall 5042, Irvine, CA 92697-3440, USA
e-mail: sjavanma@ics.uci.edu

D.W. McDonald
The Information School, University of Washington, Washington, WA, USA

R. Caruana
Microsoft Research, Redmond, WA, USA

S. Forouzan · C.V. Lopes
Bren School of Information and Computer Sciences, University of California, Irvine, CA, USA

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_11,

Keywords Vandalism detection · Wikipedia · UGC · Feature selection · Machine learning · Lasso · Cost sensitive learning · User history · Random forests

1 Introduction

Wikipedia is an open, collaboratively edited encyclopedia that is a heavily used reference source on the Internet. High visibility and the simplicity of editing almost any article have made Wikipedia a popular target for vandals. Maintaining the quality of information in Wikipedia as well as many other User Generated Content (UGC) systems requires identifying and removing vandalism.

In general, vandalism is any deliberate attempt to compromise the integrity of an online source. This covers a broad range of destructive actions such as advertising, attacks on public figures, contributing misinformation, subtle distortion through equivocation or exaggeration, phishing, altering the destination of apparently legitimate links, misleading comments on a change, faulty application of markup text—among many other forms; the range of types of vandalism is quite astonishing [19, 29].

Many UGC systems, like Wikipedia, rely on extensive manual efforts to combat vandalism. Some automated vandal fighting tools, often in the form of semi-automated bots, are being used to alleviate this laborious task. More recently, machine learning based approaches have been proposed [28]. However, a highly accurate vandalism detection approach that can be applied on the large-scale of Wikipedia is still missing. A successful approach needs to scale well, be robust in the face of widely varying levels of user participation and high user turnover—and should be able to detect vandalism in real-time.

This paper describes a low-cost and highly accurate vandalism detection model that is practical for real-time applications. This work builds on previous work on vandalism detection in Wikipedia [21]. While our more general focus is on UGC systems, we develop the model based on a Wikipedia corpus from the "Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN)" workshop. This workshop has been developing corpora and testing algorithms head-to-head since 2007 and thus provides data and benchmarks for comparing our results. In this work, we consider all the features used in the PAN competition by the participating teams and we introduce some new features, mainly user features which account for past user activity and thereby represent something of the user reputation. We build a classifier using these features and show that it performs better than prior results in the PAN competition.

Further, we try to compress the model by learning from the smallest number of features possible. First, we decrease the feature set by eliminating redundant features. Then, we take into account cost of acquisition of features relative to their individual contribution to the overall performance of the classifier. All the experiments are done based on a MapReduce paradigm, which makes our approach both efficient and scalable.

This work makes two important contributions. First, while our specific application domain is Wikipedia, the machine learning techniques are focused on leveraging low-cost features that can be generally applied to many forms of UGC. The overall approach is therefore very general and could provide a basis for a wide range of semi-automated vandalism detection tools. Second, we show the importance of a valuable set of features related to the 'reputation' of the user. By recognizing the specific value of a reputation function for vandalism detection, we provide a general approach for understanding the cost/benefit trade-off of determining reputation. Further, by illustrating the specific value of these reputation features we raise a critical challenge for large-scale networked data; What are less expensive techniques can be devised for determining user reputation from large complex networked data?

This paper is structured in the following way. We begin with a review of the relevant work on vandalism mitigation from both a user and a technical perspective. Through this we identify a number of persistent challenges for users as well as sets of features that are commonly used to develop technical solutions for vandalism detection. In subsequent sections we elaborate a relevant set of features, and use those features to train a classifier that is effective at predicting vandalism. We then apply *lasso* to learn a sparse low-cost model whose accuracy is comparable to the original classification model. We close the paper by considering some applications of the resulting model and the more general implications of our approach and findings.

2 Background

Vandalism detection has been a concern for Wikipedia since its inception. Vandalism in Wikipedia is defined as any addition, removal, or change of content in a deliberate attempt to compromise the integrity of Wikipedia. According to this broad definition, vandalism can include spamming, lobbying and destroying the edits of others. Wikipedia relies mostly on its human editors and administrators to fight vandalism; identifying instances of potential vandalism by reading a diff of the change and reverting changes that look to be a form of vandalism. But the scale of Wikipedia makes locating all vandalism very labor intensive. Tools such as Vandal Fighter, Huggle, and Twinkle are used to monitor recent changes to articles in real-time and revert those changes deemed vandalism [14].

In the following we describe two broad approaches to vandal fighting; (a) the user approach which relies mostly on tools to assist user detection, and (b) more automated approaches that generally rely on bot or other algorithms.

Viegas *et al.* [35] conducted some early work on the types of vandalism found in Wikipedia. They used a visualization technique called "History Flow" to see the various ways pages were edited and changed over time. In considering these changes they identified five types of vandalism: Mass Deletion, Offensive Copy, Phony Copy, Phony Redirection, and Idiosyncratic Copy. They also analyzed the time for repair of vandalism, which they termed "survival time". They found that the median survival time for some vandalism is quite short, on the order of minutes. However, they noted

that the mean time is skewed long, on the order of days or weeks, depending on the type of vandalism. This means there is some vandalism that goes undetected for long periods of time.

In some follow-up work, Priedhorsky *et al.* [29], considered the impact of a piece of vandalism. That is, if some vandalism lasts on a site for days, how likely is it that a user might stumble across that and be misinformed or otherwise get a wrong impression about the quality of the entire content based on a vandalized page. They developed a model based on page viewing behaviors and vandalism persistence. Their model correlates closely with the empirical results from Viegas *et al.* [35] and further illustrates that about 11 % of vandalism would last beyond 100 page views, with a small fraction of a percent lasting beyond 1000 page views.

Priedhorsky *et al.* [29] also considered the types of vandalism and how likely users were to agree on the types of vandalism. They began with the vandalism types from [35] making some small refinements and identified two additional types: Misinformation, and Partial Delete. They identified the most frequent categories of vandalism as Nonsense (Phony Copy) 53 %, Offensive 28 %, Misinformation 20 % and Partial Delete 14 %. However, they also noted that the rate of agreement among their users for some categories is somewhat low with Misinformation having the lowest level of agreement for these top four categories. This means that some of the most subtle vandalism, in the form of Misinformation, is even hard for users to agree upon.

In recent work, Geiger & Ribes [14] studied the process of editors who participate in "Recent Changes Patrolling" using Huggle. Their study raises some nice issues about the work to remove vandalism and how it is performed, illustrating how Wikipedians consider the activities of others when considering a change as potential vandalism. The study is a case study considering when an individual should be banned from the site for contributing too much content that has been deemed vandalism. In the case of the decision to ban a vandal, the effort is to understand whether the activities are intentionally designed to corrupt content and wreak havoc on the community itself. This work illustrates that there is a fair amount of vandalism which is somewhat 'routine' but that some is still difficult to detect even by people who are practiced at looking for vandalism. Another interesting insight is that most tools that support vandalism rollback do not yet categorize or provide a prediction rating for the edit being viewed. Most tools simply show the edit, the prior version, and the IP or username of the editor who made it.

A second set of approaches rely on automated bots and algorithms. This approach has generated lots of research activity, so we focus on the prior work that is most related to how we have approached the problem.

Since 2007 automated bots have been widely used to fight vandalism in Wikipedia. The most prominent of them are ClueBot and VoABotII. Like many vandalism detection tools, they use lists of regular expressions and consult databases with blocked users or IP addresses. The major drawback of these approaches is that most bots utilize static lists of obscenities and grammatical rules that are hard to maintain and, with some creativity, can be easily thwarted. A study on performance analysis of these bots in Wikipedia shows that they can detect about 30 % of the instances of vandalism [33].

Several machine learning approaches have recently been proposed that would improve vandalism detection [10, 18, 27, 33]. Comparing the performance of these approaches is difficult because of several shortcomings in early vandalism detection corpora. These early corpora were too small, they failed to account for the true distribution of vandalism among all edits, and the hand labeling of the examples had not been double-checked by different annotators. These shortcomings were resolved by the creation of a large-scale corpus for the PAN 2010 competition consisting of a week's worth of Wikipedia edits (PAN-WVC-10) [26].

The PAN 2010 corpus is comprised of 32,452 edits on 28,468 different articles. It was annotated by 753 annotators recruited from Amazon's Mechanical Turk, who cast more than 190,000 votes. Each edit in the corpus was reviewed by a minimum of three annotators. The annotator agreement was analyzed in order to determine whether each edit is a regular edit or vandalism, with 2,391 edits deemed to be vandalism. The corpus is split into a training set and a test set, which have 15,000 and 18,000 edits, respectively [28].

A survey of detecting approaches [28] shows that about 50 features were used by the 12 different teams. Features are categorized into two broad groups: (1) edit textual features; (2) edit meta information features. Edit textual features are extracted based on the text of the edit. Some features in this category are adopted from previous work on spam detection in emails or blogs. For example, "Longest character sequence" and "upper case to low case char ratio" are known to be important features for spam detection in emails [17].

Traditional spam detection systems for emails mainly rely on textual features based on the content. Most features show existence of particular words or phrases [6, 25, 31, 32]. In some cases non-textual features are extracted from meta data. For example, whether or not a message contains attachments [6].

Edit meta information mainly contains two types of features: user features and comment features. In Wikipedia when an individual makes an edit, there is the opportunity to provide a short comment on the change. As a convention, many editors will use the comment to briefly explain the rationale for the change or what type of change is being made. Comment features are extracted based on the comment related to an edit. Most teams used comment features, but two teams extensively relied on user features. User features are extracted based on the editing pattern of the user.

Identifying an editing pattern requires that some amount of state, or history, be maintained. Some forms of UGC do not maintain history as a function of the system design, but in the case of wikis user history is a given. Two teams in the PAN competition relied on user reputation features extracted from the edit revision history. Those teams placed second and third in the competition. Other approaches rely more heavily on a model of user reputation. Adler *et al.* [4] used WikiTrust to estimate user reputation. In their system, users gain reputation when their edits are preserved and they lose reputation when their edits are reverted or undone [3].

Javanmardi *et al.* [20] used user reputation features based on the reputation management system they developed earlier. Compared to [3], the model is simpler and more efficient. One reason is that it is only based on the stability of inserts. In [3],

stability of deletes and reverts are also considered. A detailed comparison between these two approaches are presented in [20].

Potthast *et al.* [28] combined the predictions submitted by the top 8 teams and developed a meta classifier based on the predictions. To learn the meta classifier, they used random forest. This improved the classification performance (ROC-AUC) of the top team significantly, from 0.91580 to 0.9569. Using a similar approach, Adler *et al.* [5] developed a meta classifier based on the predictions of three different classifiers. To evaluate the meta classifier, they merged train set and test set and reported ROC-AUC based on 10-fold cross validation. Hence, their results are not comparable with PAN competition results and our results.

In general, meta classifiers work at a macro level by aggregating results from different classifiers. However, this makes them even more complex and less practical for real-time applications. In contrast, in this study we work at the level of individual features and focus on building accurate classifiers with a minimum set of features. We show that the classification performance of the compact classifier is comparable to the meta classifier developed in [28].

3 Feature Extraction

Our feature extraction is based on the complete Wikipedia history dump, released on Jan, 2010. It turns out that 41 edits in the PAN corpus are missing from the "complete" Wikipedia dump. We use crawler4j [1] to extract the data of these missing edits. We also used additional Wikipedia SQL dumps to extract users with special access rights such as administrators and bureaucrats.

Using all these data, we extract 66 features. This feature set includes most of the features used in the PAN competition by different teams [28]. In addition, we introduce new features. Table 1 shows the features along with their definitions (note: each row might represent more than one feature).

We categorize the features into four groups. Similar to [28] we separate edit textual features and edit meta data features. Since the edit meta data features contains both user and comment features, we consider them as different groups. In addition, we add language model features as a new group. These features capture topical relevance of the inserted or deleted content and are estimated based the language model of the newly submitted revision and the background. The effectiveness of using these features for vandalism detection has been studied in [10, 24].

- *User Features*: We introduce 12 features for each user including statistical and aggregate features. We calculate these features by mining history revisions up to time T. For the purpose of this study, we consider T as 2009-11-18 which is the time-stamp of the earliest edit log in the PAN corpus. Hence, all features in this category are based solely on history data.
- *Textual Features*: We have 30 features in this category. Unlike previous work [28], in this work textual features are calculated not only based on the inserted content but also based on the deleted content. To distinguish these features we

Table 1 Feature set descriptions. Features marked with star also are extracted based on insertions and deletions

Feature	Description
User features:	
DSR, DDSR, Rep	Aggregated features representing a user's reputation
Ins Words	Total number of words inserted by a user
Del Words	Total number of words deleted by a user
Lost Words	Total number of deleted words from a user
Ins Revision	Total number of revisions a user has done insertion in
Del Revision	Total number of revisions a user has done deletion in
Ins Page	Total number of pages a user has done insertion in
Del Page	Total number of pages a user has done deletion in
User Type	User has some special rights, such an admin, a bot, or a bureaucrat
User Page	User has a user page in Wikipedia
Textual features:	
Ins Size	Number of inserted words
Del Size	Number of deleted words
Revision Size	Size difference ratio between the old and the new revision.
Blanking	The whole article has been deleted
Internal Links	Number of links added to Wikipedia articles
External Links	Number of added external links
Word Repetitions	Length of the longest word
Char Repetitions	Length of the longest repeated char sequence
Compressibility	Compression rate of the edit differences
Capitalization*	Ratio of upper case chars to lower case chars
Capitalization All*	Ratio of upper case chars to all chars
Digits*	Ratio of digits to all letters
Special Chars*	Ratio of non-alphanumeric chars to all chars
Diversity*	Length of all inserted lines to the (1/number of different chars)
Inserted Words*	Average term frequency of inserted words
Vulgarism*	Frequency of vulgar words
Bias*	Frequency (impact) of biased words
Sex*	Frequency (impact) of sex related words
Spam*	Frequency (impact) of spam related words
Pronouns*	Frequency (impact) of personal pronouns
WP*	Frequency (impact) of mark up related words
Special Words*	Aggregation of vulgarism, bias, sex, spam, pronouns, and WP ratios
Meta data features:	
Time Diff	Time interval between the submission of the old and the new revision
Category	If the automatic comment contains "category"
Early Years	If the automatic comment contains "early years"
Copyedit	If the automatic comment contains "copyedit"
Personal Life	If the automatic comment contains "personal life"
Revert	If the automatic comment contains "revert"
Revision Ordinal	Ordinal of the submitted revision
Length	Length of the comment
Reverted	If the MD5 digest of new revisions is the same as one of the old ones in window size of 10
Language model features:	
KL Distance	Kullback–Leibler distance between the old revision and the new revision
KL Distance Ins	Kullback–Leibler distance between the inserted words and the new revision
KL Distance Del	Kullback–Leibler distance between the deleted words and the new revision

use "Ins" and "Del" prefix throughout this paper. For example, *Vulgarism* shows the frequency of vulgar words. We expect insertion of vulgar words to be a signal for vandalism. Conversely, we expect deletion of such words to be a signal for legitimate edits aiming at removing vandalism.

- *Meta Data Features*: We have 22 features in this category. Most features are extracted from the comments associated with the edits. For example, we have similar textual features as in the previous category but here we extract them based on the comment. Because of descriptions we do not define them in Table 1. These features are specified by *.

  In addition to these, we introduce some new features that we extract from the automatic comments generated by Wikipedia. These comments specify which section of the article has been edited. We extract unigram, bigrams, and trigrams from these types of comments. We use feature selection on PAN train set to extract the important ones. For example, the short time interval between old and new revision might be a signal for detecting vandalism.
- *Language Model Features*: In this category we have 3 features which calculate the *Kullback–Leibler* distance (KLD) between two unigram language models. We calculate KLD between the previous and the new revision [10]. We introduce two more features: the KLD between the inserted content and the previous revision. Similarly, we calculate the KLD between the deleted content and the previous revision. Our intuition behind KLD features is that sometimes vandalism comes with some unexpected words so we expect to see sharp changes in the distance. Conversely, deleting unexpected words can be a signal for legitimate edits.

4 Learning Vandalism Detection Model

We consider vandalism detection as a binary classification problem. We map each edit in PAN corpus into a feature vector and learn the labels by mapping feature vectors onto $\{0, 1\}$, where 0 denotes legitimate edit, and 1 vandalistic edit. To learn a classifier and tune its free parameters, we use PAN train set and keep the test set untouched for final evaluation.

We use different binary classification algorithms which have been widely applied to spam detection such as Naive Bayes [32], Logistic Regression [9], and SVM [31]. We also use random forests which has been shown to be a very effective binary classifier [8]. Table 2 shows the classification performance for 3-fold cross validation on the train set and also on a test set. We use the Java implementation of these binary classifiers from Weka [16] and tune the free parameters based on cross validation. Because we use the train set to learn some of the features, and to make our results comparable to the PAN results, we do not use the test set in any way during training. The test set is only used to measure classification performance. Here we report classification performance in terms of area under the ROC curve (ROC-AUC). (The ROC-AUC metric was used in the PAN 2010 evaluation. Results for area under the

Table 2 Classification performance for the binary classifiers

Algorithm	ROC-AUC (CV)	ROC-AUC (Test)
Naive Bayes	0.9482 ± 0.0057	0.9068
Logistic Regression	0.9494 ± 0.0070	0.9128
SVM (LibSVM, RBF Kernel, Gamma = 0.15, c = 11)	0.9537 ± 0.0007	0.9202
Random Forest (1000 trees)	0.9739 ± 0.0024	0.9553

precision-recall curve are similar.) In all our experiments random forests outperformed the other classifiers.

4.1 Vandalism Detection Using Random Forest

Random forest is not widely used for spam detection because it is known to be expensive in terms of time and space [34]. In this section we explain how we can train random forest classifiers efficiently and gain high classification performance at the same time.

Statistical analysis of Wikipedia edits show that roughly 7 % of edits are vandalistic [20], which is consistent with the vandalism ratio in PAN corpus. Given this, we need to use machine learning algorithms which are robust to imbalanced data, such as random forests. In addition, random forests are a suitable option for datasets with missing data [7]. The PAN data set is an imbalanced dataset and some features such as the user group features are sometimes missing; For 4 % of users we do not have any user group information at all. This suggests that a random forest model could have benefits over other techniques that may not deal as well with imbalanced data or missing features. One advantage of the ROC-AUC metric is that it is fairly insensitive to imbalanced data.

To learn a random forest classifier, we need to tune two free parameters: the number of trees in the model and the random number of features considered in each split. Our experiments show that classification performance is sensitive to the former but not to the latter. This result is consistent with Breiman's observation [7] on the insensitivity of random forests to the number of features considered in each split.

To tune the number of trees, we partition the train set into three folds and use 3-fold cross validation. Using three folds allowed us to keep a reasonably large number of vandalized cases in each training set (around 600). To find the optimal value for the number of trees, we need to sweep a large range of values. Hence, we need to design an efficient process for this purpose.

For each fold, we create a pool of $N = 10{,}000$ trees, each trained on a random sample of the training data in that fold. Then we use this pool for creating random forests of different sizes. For example, to create a random forest with 20 trees, we randomly select 20 trees from this pool of N trees. However, since this random selection can be done in $C(N, 20)$ different ways, each combination may result in

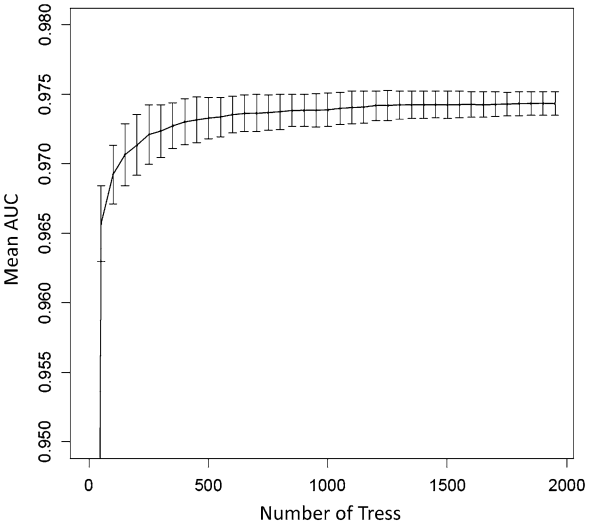


Fig. 1 The effect of number of trees on AUC mean and standard deviation

a different AUC. We repeat the random selection of trees $r = 50$ times in order to calculate the mean and variance of the $F \times r$ results (where F is the number of folds).

The advantage of this approach is that we can calculate the mean and variance of AUC very efficiently for forests with different sizes without the need to train a huge number of trees. Otherwise, to report the mean and variance of AUC for random forests of size $k = 1 \rightarrow T$, we would need to train $r + 2 \times r + 3 \times r + \cdots + T \times r = r * T(T + 1)/2$ trees for each fold, which is 10^8 trees. Using our approach we only need to train N trees per fold (in our experiments we used $N = 5 \times T$).

Figure 1 shows the mean of AUC as a function of number of trees in the model. As more trees are added to the model, mean of AUC increases and the variance decreases. The mean of AUC does not improve significantly after more than about 500 trees in the random forest but the variance continues decreasing. It should be emphasized that models with smaller variance are more stable and therefore more predictable in test environments. Although more trees may result in slightly better AUC values, we decide to set the number of trees at 1000 to have a balance between classification performance and model complexity. More complex models with more trees would require more time for prediction in a real-time application. Given this, the AUC on for 3-fold cross validation on the train set is 0.9739 ± 0.0024. The AUC value on PAN test set is **0.9553**. This result is significantly higher than the best AUC reported to the PAN competition which was 0.9218 [28].

5 Feature Selection

The classifier mentioned in the previous section makes its decision based on 66 features which fall into four logically different groups. However, computing and

Table 3 The drop in mean AUC when we eliminate a feature group. Results on train set are for 3-fold cross validation

Dropped group	Train set	Test set
User	−3.8 %	−6.6 %
Textual	−1.8 %	−1.5 %
Meta data	−0.4 %	−0.3 %
Language model	−0.2 %	−0.3 %

updating each of these features imposes significant off-line cost at run-time. For example, computing and updating features in the user group requires the tracking of all edits done by each individual. Maintaining a system that updates data for computing these features would come at a cost for the wiki. Some other features like textual features are computed after submission of a new edit and the vandalism detection system should be able to compute them in real-time.

In this section, we report the results of our experiments in finding a minimum set of features whose classification performance is almost as good as the one with all 66 features. In other words, we try to detect and eliminate redundant or unnecessary features. We consider two types of redundant or unnecessary features: (a) features that are not informative and do not help in discriminating legitimate and vandalistic content; (b) features correlated with some other features so that once one of them is selected, adding others does not add any new discriminating power to the model.

In order to detect and eliminate redundant features, we perform two sets of experiments. First, in Sect. 5.1 we study the contribution of groups of features as a whole unit to examine if any of the four groups can be eliminated without a significant drop in AUC. Then in Sect. 5.2 we study the contribution of each feature individually and use the results of this analysis to eliminate redundant features.

Given the large number of experiments needed for this study, we use the Amazon MapReduce cluster to run them in parallel. In our implementation, each mapper receives specific config information and trains a classifier for that configuration. Then reducers aggregate the AUC results for different folds and report the mean AUC for the different configs.

5.1 Eliminating Groups of Features

For each group of features, we train a classifier without the features in that group. This will show us the drop in AUC when this group is ignored. Table 3 shows the results. These results indicate that removing features in user group and textual group results in large drops in AUC, while the drop in AUC for the meta data and the language model groups is much less.

Based on these results we can not infer that features in the Meta data and LM group are not informative. The only conclusion is that once we have both user and textual features in our feature set, adding meta data and language model features

Table 4 The mean AUC when we only use features in one group

Selected group	Train set	Test set
User	0.9399	0.9225
Textual	0.9001	0.8400
Meta data	0.7019	0.6924
LM	0.7271	0.6986
Baseline (all groups)	0.9739	0.9553

does not add substantial additional discriminating power to the model. Table 4 supports this interpretation: when we only use meta data or language model features the AUC value is much higher that of a random classifier (0.50). The single group of features that results in the highest AUC (0.9399) is the user group.

5.2 Eliminating Individual Features

In Sect. 5.1 we showed that all the four groups contain informative features. In this section, we attempt to find the smallest feature set whose AUC is comparable to the AUC of a classifier with 66 features. To this aim we do feature selection.

There are three different approaches for performing feature selection, univariate feature analysis, wrapper-based methods, and proxy methods [15, 22]. Univariate feature analysis methods such as Information Gain or Chi-Square evaluate features independently in order to estimate their importance. Because these methods evaluate the utility of each feature in isolation, they are unable to detect and benefit from correlations among the features and often yield unnecessarily large feature sets.

In wrapper-based methods one wraps a feature selection process such as forward stepwise selection or backwards stepwise elimination around the learning algorithm that will be used to train the final model so that the feature selection process can find a small set of features that works well with that learning method. Because the contribution of each feature is evaluated in the context of the other features used in the model, wrapper-based methods are very effective at eliminating features whose contribution to the model is redundant with other features already in the model, and thus often yield the smallest feature sets. The main difficulty with the wrapper approach is that it can be prohibitively expensive to wrap feature selection around expensive learning methods such as random forests. For this reason, proxy feature selection methods are often used. In proxy methods, feature selection is performed for a simpler, more computationally tractable model class such as linear or logistic regression models, and the features found to be most important for the simpler model class is then used in the more complex, more expensive model class (in this paper random forests). Because proxy methods do not take the specific learning algorithm into account, they do not always find as compact a set of features as wrapper methods. However, in practice there usually is strong overlap between the features that are best for a simpler model and those that are effective in more complex models, so

proxy methods often yield feature sets that are almost as small as those returned by wrapper methods.

Because the model class that appears to perform best on vandalism detection is computationally expensive (random forests), in this paper we consider only proxy feature selection methods. We need a feature selection algorithm that is efficient and which considers correlations between features and is able to detect and eliminate redundant features effectively. We use Lasso Logistic Regression (Least Absolute Shrinkage and Selection Operator for Logistic Regression) [13] for this purpose. This fits a regularized logistic regression model to features in such a way that the final model has a sparse solution in the feature space. Thus, the weight of redundant features in the final model would be zero. It means that we can remove these features from the model with no significant change in the classification performance. For this, we use the Logistic Regression Lasso implemented in glmnet package in R [13].

Lasso for logistic regression has a regularization parameter, λ, that is a trade off between sparsity of the model and its classification performance. Lower values for λ result in more relaxation of the regularization constraint which allows more features to have non-zero weights. The R package, glmnet [13], uses regularized maximum (binomial) likelihood to fit this regularized logistic regression model to the data. The problem can be formalized by the following:

$$\max_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[l(\beta_0, \beta) - \lambda \|\beta\|_1 \right] \tag{1}$$

where

$$l(\beta_0, \beta) = \frac{1}{N} \sum_{i=1}^{N} y_i \log\big(p(x_i)\big) + (1 - y_i) \log\big(1 - p(x_i)\big) \tag{2}$$

and

$$p(x_i) = \frac{1}{1 + \exp(\beta_0 + x_i^T \beta)} \tag{3}$$

Table 5 shows the features selected for different values of λ. For $\lambda = 0.0716$ only one feature is selected. This means that according to lasso if we want to make the classification model based on only one feature, "Ins Special Words" would be our best choice. As we decrease the value of λ, more features are selected according to their importance. The second most important feature is "DDSR". The last column in Table 5 shows the value of the AUC on the PAN test set for classifiers trained on the selected features. As the number of selected features increases, we see a higher value for AUC but the cost of computing and updating the features would also increase.[1]

We use 3-fold cross validation on the PAN train set to pick the largest value for λ, where the drop in AUC is not statistically significant. The result was $\lambda = 0.0030$

[1]The LASSO method does not necessarily yield a monotonically increasing set of features; it is possible that as λ is decreased some features that were in the set for larger λs might be removed from the set as other feature replace them.

Table 5 Feature selection using lasso. Parameter λ determines a trade off between number of selected features and AUC of the classifier. Smaller values of λ allow more features be selected and result in models with higher performance

λ	Selected features	AUC on PAN test set
0.0716	Ins Special Words	0.5974
0.0594	DDSR, Ins Special Words	0.8614
0.0373	DDSR, Rep, Ins Special Words	0.8965
0.0340	DDSR, Rep, User Page, Ins Digits, Ins Special Words	0.9074
0.0310	DDSR, Rep, User Page, Ins Digits, Ins Vulgarism, Ins Special Words	0.9090
0.0257	DDSR, User Page, KLDNew2OLD, Ins Digits, Ins Vulgarism, Ins Special Words	0.9197
⋮	⋮	⋮
0.0030	DDSR, Del Words, User Type, User Page, Copyedit, Personal Life, Revision Ordinal, Comment Length, KLDNew2OLD, Blanking, Ins Internal Link, Ins External Link, Longest Inserted Word, Ins Longest Char Repetitions, Ins Compressibility, Ins Capitalization, Ins Digits, Ins Special Chars, Ins Vulgarism, Ins Bias, Ins Sex, Ins Pronouns, Ins WP, Ins Special Words, Del Bias, Ins Digits, Comment Special Chars Comment Spam	0.9505
⋮	⋮	⋮

which leads to selection of 28 features. Table 5 shows a list of the selected features. The AUC for this feature set on PAN test set is 0.9505. This feature set only includes less than half of the original features but the drop in AUC is only 0.005.

In this sparse feature set we have features from all groups. For example, "DDSR", "Del Words", "User Type", and "User Page" are selected from the user group. If we follow the Lasso path, features get added and eliminated as λ decreases. For example, "Rep" is selected as the third important feature, but because of its correlation with other selected features it is eliminated from the feature set later. Interestingly, "Rep" is computationally more expensive than "DDSR".

We have 17 features from the textual group. These features are also widely used for spam detection in other domains such as emails or blogs [17]. For example, "Ins Longest Char Repetitions" shows whether a user has inserted a long sequence of the same character which can be a good indicator of vandalism. There are also some textual features which are unique to Wikipedia. For example, "Del Bias" shows that the user has deleted words which represent bias and therefore is a good indicator of legitimate edit.

We have 6 features selected from the meta data group. For example, "Comment Length" or "Comment Special Chars" are selected as important features. "Personal Life" is another important feature. It shows whether the edit is made in the "Personal Life" section of a biography article. We have observed that this section of biography articles is more often vandalized and therefore this feature can be an important

signal. This observation is consistent with Wikipedia statistics which shows high vandalism ratio in biography articles [2]. Given that Wikipedia automatically adds the name of an edited section to the comment associated to each edit, we can extract this feature from comments.

The only feature that is selected from the Language Model group is "KLD-New2OLD". The goal of this feature is to detect sharp linguistic distance between the new and the previous revision which can be an important signal for vandalism detection.

6 Cost Sensitive Feature Selection

In Sect. 5 we focused on features selection in order to find a small subset of available features that would encapsulate all useful information for the task at hand. This was useful to eliminate redundant features and increase the interpretability of the selected features. However, in many applications, it is also useful to take into account the amount of effort required to compute the features. The importance of feature cost is ignored by many machine learning methods [12].

The general setting of cost sensitive learning is consists of designing classification models while taking into account the cost involved in the entire decision process. This includes the cost of data acquisition, the cost of labeling the training samples and the cost of making different decision errors [30]. In this work we focus on the first case where there are different costs to acquire different features. For example, to calculate value of the feature "Rep" we need to process all history revisions and track contributions of each individual user; we also need to track how other users have edited their contributions [20]. Some other features like "Comment Length" are easier to compute and only need a little light text processing.

To address this issue here we try to take into account the cost of acquiring different groups of features during the learning process. We are interested in cases where well-performing cheap features are preferred over expensive ones with slight difference in overall classification results. For this purpose, we study two different scenarios. In the first scenario features are selected based on their corresponding costs.

6.1 Cost Sensitive Lasso

In this section we use regularized logistic regression based on lasso to implement cost sensitive feature selection. Unlike in Sect. 5.2 where we considered a fixed penalty λ for all features, here we assign higher penalties to the computationally more expensive features. We use glmnet package in R [13] which allows different penalties λ_j for each of the variables via a penalty scaling parameter $\gamma_j \geq 0$. If $\gamma_j > 0$, then the penalty applied to β_j is $\lambda_j = \lambda\gamma_j$. If $\gamma_j = 0$, that variable does not

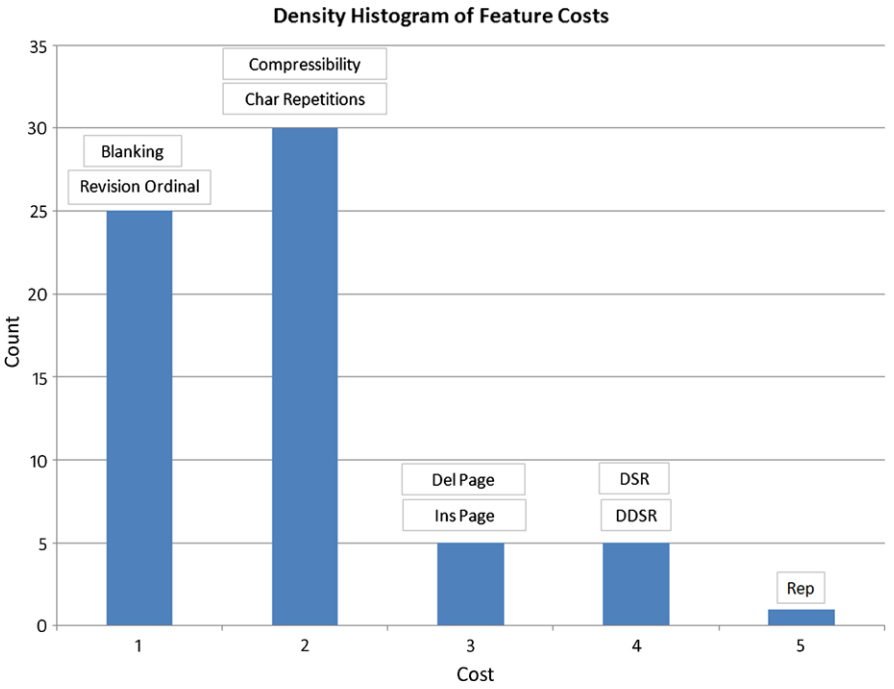


Fig. 2 Histogram of cost of acquisition of features

get penalized, and enters the model unrestricted at the first step and remains in the model. Here we set the values of each γ_j relative to the cost of acquisition of the feature; in the previous scenario γ_j was set to 1 for all the features.

Here the costs assigned to features vary between 1 and 5; 1 is for the least computationally expensive feature and 5 is for the most computationally expensive one. Figure 2 shows the histogram of cost values in our feature set along with some examples in each category.

Considering these features' costs we run lasso for logistic regression. Table 6 shows how features are selected in the lasso path along with the value of AUC. "User Page" whose cost is 1 is the first feature that is selected. As features are added to the feature set AUC tends to increase. Seven features were not contributing to AUC for any value of λ and are left out from Table 6. Note that there is a significant jump in AUC when "DDSR" is added to the feature set, from 0.8245 to 0.9065. This feature is among the most computationally expensive features but because of its importance it has been selected in the first steps of feature selection. This result is consistent with the results in Table 5 where "DDSR" was selected as the second most important feature. In that case we assumed similar cost for all the features while here we assigned high cost to this feature. The fact that this feature is selected in early in the cost-sensitive feature selection process despite it's high cost, and the large gain in AUC, suggests the importance of this user feature.

Table 6 Cost sensitive feature selection using lasso. Column 4 shows the features selected via cost-sensitive lasso on the training set and column 5 shows AUC of random forest classifier on the test data

λ	Feature set size	Cost	Newly selected features	AUC on PAN test set
0.0882	1	1	User Page	0.7129
0.0882	2	2	Ins Special Words	0.7581
0.0476	3	1	Revision Ordinal	0.6832
0.0447	4	2	Ins Character Repetition	0.7087
0.0396	5	1	User Type	0.7266
0.0291	6	2	Ins Vulgarism	0.7235
0.0282	7	2	Ins Digits	0.8245
0.0265	8	4	DDSR	0.9065
0.0257	9	2	KL Distance	0.9299
0.0242	10	2	Ins Bias	0.9305
⋮	⋮	⋮	⋮	⋮
0.0021	32	2	Del Special Words	0.9486
0.0019	33	3	Ins Revision	0.9516
⋮	⋮	⋮	⋮	⋮
0.0014	58	4	Del Words	0.9543
0.0001	59	2	KL Distance Ins	0.9553

After selecting 33 features, AUC is comparable to the AUC of a classifier with 66 features. This means that having about half of the features in the feature set the classification performance is roughly the same. However, in contrast to Sect. 5.2, here we are training classifiers using a similar number of features *but they are less computationally expensive features*.

6.2 Cost Sensitive Group Lasso

In this section we do cost sensitive feature selection but also take into account the data sources the features are extracted from. We categorize the features into five groups; where features in one group are captured in the same way and based on the same data source. For example, when we process text of history revisions and extract word ownerships, we have the data to calculate features like "DSR", "DDSR", and "Rep". In other words, when we pay the cost of processing text of history revisions to calculate a feature like "Rep" then the cost of calculating a new feature like "DSR" will be zero. Group Lasso [23] captures this notion and provides a way for cost sensitive feature selection. When a feature from a group is selected, other features of that group will be selected automatically.

Table 7 Cost sensitive feature selection using group lasso

λ	Feature set size	Selected group	AUC
350	1	Group 5	0.7129
330	2	Group 5 + 4	0.7581
300	13	Group 5 + 4 + 1	0.9230
50	45	Group 5 + 4 + 1 + 2	0.9519
10	66	Group 5 + 4 + 1 + 2 + 3	0.9555

We acquire features in five different ways: (1) processing text of all history revisions of articles; (2) processing the text of current revision of articles; (3) processing meta data of current revision of articles; (4) processing sql dumps; and (5) crawling Wikipedia user pages. We process the text of history revisions to track contributions of users in order to acquire some features like "DSR","DDSR", and "Rep". For some other features like "Time Diff", "Category", or "Comment Length" we do not need to process the entire history dump and processing the meta data of the current revisions is enough. We use Wikipedia sql dumps to acquire features like "User Type". We also crawl Wikipedia user pages in order to calculate the value of features like "User Page".

To be able to incorporate group information into our feature selection process, we use group lasso as implemented in the package "grplasso" [23] in R. "grplasso" estimates the weights of a logistic function, considering the group information for different features. Logistic group lasso estimator β can be computed by the maximizing the following convex function:

$$\max_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[l(\beta_0, \beta) - \lambda \sum_{g=1}^{G} s(df_g)\|\beta_g\|_2 \right] \tag{4}$$

Here, the loss function l is defined as in (2) and β_g is the parameter vector corresponding to the gth group of the predictors. The function $s(.)$ is used to rescale the penalty with respect to the dimensionality of the parameter vector β_g, the default value of which is set to $s(df_g) = df_g^{1/2}$ to ensure that the penalty term is of the order of the number of parameters df_g.

Table 7 shows the result. The first group selected is Group 5 which has one feature "User Page". Then Group 4 is added which has one feature, "User Type". Group 1 is added in the third step which has 10 features. All these three groups contain user related features. When Group 3 is added we see a significant jump in the value of AUC, about 16 %. It shows the importance of user group features.

The next group which is selected is group 2 which mostly consists of textual features. Adding this group, we see some improvements in the AUC. The last selected group is Group 3 which contains meta data features. The small increase in the value of AUC shows that adding this group of features does not improve AUC significantly.

According to group lasso, user features are the most important features, confirming the results of feature selection experiments from the previous sections. Although

these features are computationally expensive to acquire compared to other features, they are selected in early steps of the lasso path because of their significant contribution to classification performance. This is very interesting because in traditional spam detection systems for web content or emails there are generally no user related features. Maintaining a history of user activity in UGC systems is becoming more common. Many wikis maintain history which makes computing user features possible. But that still opens the question of how to maintain user history in a way that computing user level features is fast and incremental.

7 Discussion and Conclusion

In this paper we described a machine learning approach to detect vandalism in User Generated Content (UGC) systems. Our corpus is Wikipedia, but the approach is general and can be applied to many forms of UGC.

Based on the validated corpus of Wikipedia edits from the PAN competition we trained and tested several binary classifiers using learning methods that have been widely used for spam detection: Naive Bayes, Logistic Regression, and SVMs. We also trained models using random forests which have not been widely used for spam detection. Interestingly the results show that the random forests significantly outperform the other three types of classifiers on this problem. An additional benefit of random forests is that they are robust to missing and unbalanced data which is a common characteristic of vandalism/spam data sets.

The common practice when training models for spam detection has been to train the models using all available features. Because our goal is to develop models that are small and fast enough to be used in a real-time vandalism detection tool, we used feature selection to eliminate redundant features so that the computational complexity of the final model is as small as possible while retaining high accuracy. Some of the features that proved to be most informative require mining user action histories. Computing the value of these features can be expensive. Because of this, we considered both traditional feature selection, as well as cost sensitive feature selection that takes into account the cost of acquisition of each feature. We compressed the learned model by estimating the contribution of different features and feature groups to the random forest model. Across the four groups of features we found that each group contained some important features, but the user features representing the history of user contributions are most important, so we could not ignore this group of features. This motivated a focus on individual features to determine which specific features (instead of groups of features) contributed most significantly to the model. Using lasso, we found a minimum set of features whose classification performance is almost as good as the one with all 66 features. Furthermore, we did cost sensitive feature selection to force learning to prefer well-performing cheap features over more expensive features that yield only slight advantage in classification results. In combination, these techniques help us train a compact, sparse vandalism detection model that scales well and executes in real-time.

Using feature selection (lasso), we showed that the compressed model with 28 features has accuracy of 0.9505; which is comparable to the accuracy of the model with 66 features. Using cost sensitive feature selection (cost sensitive lasso), the accuracy is 0.9493 for a model with 28 features. These features are less expensive to be acquired compared to the features selected in lasso. Using group lasso we categorized features based on the source of the acquisition and showed the importance of each group of features. In one of our recent work, we studied the importance of each group of features in detecting various type of vandalism. For example, we showed that for vandalism with the type of misinformation or spam, user features are the most important features, while for detecting mass delete textual features are the best features [19].

The methods we use are not specific to wikis and can be applied to other UGC systems such as blogs, twitter, or social bookmarking systems. The new features we created such as the user reputation features in form of DDSR and usage of special characters in the text were some of the most important features in the random forest model and can be easily extracted in other UGCs. Usage of special characters has been widely used for spam detection in emails or blog comments, but user features such as DDSR which measure the survivability of the content contributed by a user, generally have not been used. This notion of survivability can be translated to other domains as well. For example, for Twitter we can see how often and how fast a tweet gets re-tweeted. Similarly, in Facebook, we can look at patterns of sharing and propagation to measure survivability.

One of the main applications of the methods we develop is for end-user tools that support vandalism detection and reversion. Few of the tools currently in use predict which of the viewed edits is (or is not) vandalism. This is because most predictive models have either been not accurate enough, or have been too slow to use in real-time. The models developed in this paper are more accurate and faster than previously developed models for this task.

Another possible application of this work is as part of a change awareness tool. Most wikis support a watch list. When a user puts a specific wiki page on their personal watch list they are indicating interest in changes that are made to that page. When another user changes the page, email is sent to notify everyone who is watching that page. The amount of email might not be a problem if a user is interested in only a few pages. However, when a user is interested in many pages this can result in an unmanageable flood of email. Our model could be incorporated into the watch list mechanism to allow users to specify a vandalism threshold. That is, a user might want to set different vandalism thresholds for different pages so that they are alerted only when the prediction for a change exceeds the specific threshold they set for that page. This would allow a user to monitor a broader span of pages with reduced workload.

One open issue in our approach is the initial generation of features. In our specific application, we were aided by prior work on Wikipedia. We also identified user features based on user reputation as particularly important for detecting vandalism and suggested modifications of these for application in blog spam detection and for detecting interestingness in systems like Twitter [11]. The importance of user history

features is perhaps not surprising, but raises some challenges for UGC systems. User level features require the compilation of user action histories. In some systems this is not kept by default. One impact of our results suggest that systems which currently do not store user history should reconsider this design decision.

The value of these user level features points out a critical challenge for creating, managing, and analyzing large-scale networked data. First, social and web networks provide a broader range of user level features not considered in our model. These relational features can be considered either individually or in aggregate form. As participation in a UGC system grows the size and cost of computing those reputation (relational) features will grow. This suggests that low-cost, ego-centric and incremental approaches to calculating reputation will be valuable. Our approach provides a technique for analyzing the cost/value trade-off of maintaining user level features.

Vandalism will continue to be a challenge for all UGC systems. What constitutes vandalism will always be something that is in the eye of the beholder. As such, the community of users contributing to UGC systems will always be the final arbiters of what should stay and what should go. Machine learning-based tools provide a promising approach that is accurate, tunable, and maintainable for assessing edits in UGC systems.

Acknowledgements Authors would like to thank Prof. Robert Tibshirani and Prof. Alexander Ihler for their comments on using lasso for cost sensitive feature selection, and also Martin Potthast for his support in making the PAN data set available to us. In addition, authors would like to thank Amazon.com for a research grant that allowed us to use their MapReduce cluster. This work has been also partially supported by NSF grant OCI-074806.

References

1. crawler4j: A fast crawler in java. http://crawler4j.googlecode.com/
2. Wikipedia article on biography controversy. http://en.wikipedia.org/wiki/Wikipedia_biography_controversy
3. Adler BT, de Alfaro L (2007) A content-driven reputation system for the wikipedia. In: WWW '07: proceedings of the 16th international conference on world wide web. ACM, New York, pp 261–270
4. Adler B, de Alfaro L, Pve I (2010) Detecting wikipedia vandalism using wikitrust. Tech. rep., PAN lab report, CLEF (Conference on multilingual and multimodal information access evaluation). institute.lanl.gov/isti/issdm/papers/vandalism-adler-report.pdf
5. Adler BT, de Alfaro L, Mola-Velasco SM, Rosso P, West AG (2011) Wikipedia vandalism detection: combining natural language, metadata, and reputation features. In: Proceedings of computational linguistics and intelligent text processing, CICLing'11, pp 266–276
6. Androutsopoulos I, Paliouras G, Karkaletsis V, Sakkis G, Spyropoulos CD, Stamatopoulos P (2000) Learning to filter spam e-mail: a comparison of a naive bayesian and a memory-based approach. In: Proceedings of CoRR, pp 1–13
7. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
8. Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on machine learning, ICML'06. ACM, New York, pp 161–168

9. Chang Mw, Yih Wt, Meek C (2008) Partitioned logistic regression for spam filtering. In: Proceeding of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'08. ACM, New York, pp 97–105
10. Chin Sc, Srinivasan P, Street WN, Eichmann D (2010) Detecting wikipedia vandalism with active learning and statistical language models. In: Fourth workshop on information credibility on the web, WICOW 2010
11. Dong A, Zhang R, Kolari P, Bai J, Diaz F, Chang Y, Zheng Z, Zha H (2010) Time is of the essence: improving recency ranking using twitter data. In: World wide web conference series, pp 331–340
12. Forman G (2003) An extensive empirical study of feature selection metrics for text classification. J Mach Learn Res 3:1289–1305
13. Friedman JH, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. J Stat Softw 33(1):1–22. http://www.jstatsoft.org/v33/i01
14. Geiger RS, Ribes D (2010) The work of sustaining order in wikipedia: the banning of a vandal. In: Proceedings of the 2010 ACM conference on computer supported cooperative work, CSCW'10. ACM, New York, pp 117–126
15. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
16. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. ACM SIGKDD Explor Newsl 11(1):10–18. doi:10.1145/1656274.1656278
17. Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning. Springer series in statistics. Springer, New York
18. Itakura KY, Clarke CLA (2009) Using dynamic Markov compression to detect vandalism in the wikipedia. In: Proceedings of the 32nd international ACM SIGIR conference on research and development in information retrieval, SIGIR'09. ACM, New York, pp 822–823
19. Javanmardi S (2011) Measuring content quality in user generated content systems: a machine learning approach. Doctoral dissertation, University of California, Irvine, CA, Chapter 7
20. Javanmardi S, Lopes C, Baldi P (2010) Modeling user reputation in wikipedia. J Stat Anal Data Min 3(2):126–139
21. Javanmardi S, McDonald DW, Lopes CV (2011) Vandalism detection in wikipedia: a high-performing, feature–rich model and its reduction through lasso. In: Proceedings of the 7th international symposium on wikis and open collaboration, WikiSym'11. ACM, New York, pp 82–90
22. Kohavi R, John G (1997) Wrappers for feature selection. Artif Intell 97:273–324
23. Meier L, van de Geer S, Bühlmann P (2008) The group lasso for logistic regression. J R Stat Soc, Ser B, Stat Methodol 70(1):53–71
24. Mishne G, Carmel D, Lempel R (2005) Blocking blog spam with language model disagreement. In: AIRWeb, pp 1–6
25. Narisawa K, Ikeda D, Yamada Y, Takeda M (2006) Detecting blog spams using the vocabulary size of all substrings in their copies. In: Proceedings of the 3rd annual workshop on weblogging ecosystem
26. Potthast M (2010) Crowdsourcing a wikipedia vandalism corpus. In: Proceeding of the 33rd international ACM SIGIR conference on research and development in information retrieval, SIGIR'10. ACM, New York, pp 789–790
27. Potthast M, Stein B, Gerling R (2008) Automatic vandalism detection in wikipedia. In: Macdonald C, Ounis I, Plachouras V, Ruthven I, White RW (eds) ECIR. Lecture Notes in Computer Science, vol 4956. Springer, Heidelberg, pp 663–668
28. Potthast M, Stein B, Holfeld T (2010) Overview of the 1st international competition on wikipedia. In: CLEF'2010
29. Priedhorsky R, Chen J, Lam S, Panciera K, Terveen L, Riedl J (2007) Creating, destroying, and restoring value in wikipedia. In: GROUP'07: proceedings of the 2007 international ACM conference on supporting group work. ACM, New York, pp 259–268

30. Santos-Rodriguez R, Garcia-Garcia D (2010) Cost-sensitive feature selection based on the set covering machine. In: International conference on data mining workshops, pp 740–746. doi:10.1109/ICDMW.2010.92
31. Sculley D, Wachman GM (2007) Relaxed online svms for spam filtering. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR'07. ACM, New York, pp 415–422
32. Seewald AK (2007) An evaluation of naive Bayes variants in content-based learning for spam filtering. Intell Data Anal 11:497–524
33. Smets K, Goethals B, Verdonk B (2008) Automatic vandalism detection in wikipedia: towards a machine learning approach. In: Proceedings of the association for the advancement of artificial intelligence (AAAI) workshop on wikipedia and artificial intelligence: an evolving synergy, WikiAI08. AAAI Press, Menlo Park, pp 43–48
34. Tang Y, Krasser S, He Y, Yang W, Alperovitch D (2008) Support vector machines and random forests modeling for spam senders behavior analysis. In: Global telecommunications conference, pp 2174–2178
35. Viegas FB, Wattenberg M, Dave K (2004) Studying cooperation and conflict between authors with history flow visualizations. In: CHI'04: proceedings of the SIGCHI conference on human factors in computing systems. ACM, New York, pp 575–582

Perspective on Measurement Metrics for Community Detection Algorithms

Yang Yang, Yizhou Sun, Saurav Pandit, Nitesh V. Chawla, and Jiawei Han

Abstract Community detection or cluster detection in networks is often at the core of mining network data. Whereas the problem is well-studied, given the scale and complexity of modern day social networks, detecting "reasonable" communities is often a hard problem. Since the first use of k-means algorithm in 1960s, many community detection algorithms have been presented—most of which are developed with specific goals in mind and the idea of detecting meaningful communities varies widely from one algorithm to another.

As the number of clustering algorithms grows, so does the number of metrics on how to measure them. Algorithms are often reduced to optimizing the value of an objective function such as modularity and internal density. Some of these metrics rely on ground-truth, some do not. In this chapter we study these algorithms and aim to find whether these optimization based measurements are consistent with the real performance of community detection algorithm. Seven representative algorithms are compared under various performance metrics, and on various real world social networks.

The difficulties of measuring community detection algorithms are mostly due to the unavailability of ground-truth information, and then objective functions, such

Y. Yang (✉) · S. Pandit · N.V. Chawla
Department of Computer Science & Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
e-mail: yyang1@nd.edu

S. Pandit
e-mail: spandit@nd.edu

N.V. Chawla
e-mail: nchawla@nd.edu

Y. Sun · J. Han
Department of Computer Science, University of Illinois, Urbana and Champaign, Urbana, IL 61801, USA

Y. Sun
e-mail: sun22@uiuc.edu

J. Han
e-mail: hanj@uiuc.edu

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_12,

as modularity, are used as substitutes. The benchmark networks that simulate real world networks with planted community structure are introduced to tackle the unavailability of ground-truth information, however whether the simulation is precise and useful has not been verified. In this chapter we present the performance of community detection algorithms on real world networks and their corresponding benchmark networks, which are designed to demonstrate the differences between real world networks and benchmark networks.

Keywords Social network · Community detection · Objective functions · Benchmark network · Measurements

1 Introduction

Community detection algorithms attract a great deal of attention from researchers in computer science [17, 18], especially in the area of data mining, and becoming more and more important due to the rapid proliferation of social networks, such as Facebook, the "blogosphere" or even cellular phone communication networks. However, how to effectively measure the performance of community detection algorithms remains a problem without consensus. Currently many objective functions are used to evaluate the quality of detected communities, but whether these objective functions are good approximations of performance are not yet clear. We propose to compare community detection algorithms under various performance metrics, and on several social networks to explore whether current objective functions are consistent with the "ground-truth" of social network datasets. Another important purpose of our survey is to take a closer look at whether a consensus can at all be reached or whether different community detection algorithms are effective on different networks.

In order to conduct appropriate experiments, we divide the community detection algorithms into two categories based on whether the social network is heterogeneous or homogeneous. Additionally considering the heuristics or philosophy employed by community detection algorithms, some of the heuristics could be formalized into objective functions, e.g. modularity [16, 22] and partition density [1]. Then the clustering problem virtually reduces to by maximizing or minimizing these objective functions. However in some other algorithms, heuristics are hard to be abstracted by objective functions, such as RankClus [21].

As for performance metrics, they can also be classified into two categories according to whether their evaluations rely on ground-truth or not. We are using the metrics listed in Table 1, which are used frequently as performance metrics.

Besides those performance metrics we discussed above, Andrea Lancichinetti et al. [10] proposed to use benchmark networks with built-in communities to evaluate the performance of community detection algorithms, this method is also involved in our comparisons. Our objective of experiments on benchmark networks is to an-

Table 1 Performance metrics of community detection algorithms

Metrics	Based on ground-truth	Not based on ground-truth			
	Rand index	Internal density	Conductance	Cut ratio	Modularity
Equation	$\frac{SS+DD}{SS+SD+DS+DD}$	$\frac{2m_k}{n_k(n_k-1)}$	$1-\frac{l_k}{(2m_k+l_k)}$	$1-\frac{l_k}{n_k(n_k-1)}$	$\sum_{k=1}^{K} \text{mod}_k/2M$
Comments	The first character of each variable states whether two nodes are from the same ground-truth class, and similarly the second character of each variable represents whether they are classified together by the algorithm.	This is the internal density of links within the community C_k [11].	This is the fraction of total edge number pointing outside the community [11].	This is the fraction of all possible edges leaving the community structure [11].	This states the quality of communities [11].

For all these metrics, high score indicates better quality

swer the question that whether these simulated benchmark networks are reliable to measure the performance of community detection algorithms.

The remainder of the chapter is organized as follows. We introduce the preliminary information and related work in Sect. 2. Section 3 discusses our observations on experimental results collected from small networks. The discussion of large-scale networks results are presented in Sect. 4. We provide analysis of benchmark networks in topological perspective, and present associated experimental results in Sect. 5. The conclusion of our study is drawn in Sect. 6.

2 Related Work

Here we survey related work and discuss preliminary information for our work.

2.1 Community Detection Algorithms

A great deal of work has been devoted to finding communities in networks, and much of this has been designed to formalizing heuristic that a community is a set of nodes that has more intra connections than inter connections. The algorithms used in our survey are selected according to categories described in Sect. 1. We try to select a set of community detection algorithms, which are comprehensive and representa-

Table 2 Community detection algorithms

Algorithm	RankClus [21]			LinkCommunity [1]			LineGraph [6]		
	Formalization	Heter	Homo	Formalization	Heter	Homo	Formalization	Heter	Homo
Properties	No	Yes	N/A	Yes	Yes	Yes	Depends	Yes	Yes

Algorithm	Walktrap [16]			SPICi [9]			Betweenness [8]		
	Formalization	Heter	Homo	Formalization	Heter	Homo	Formalization	Heter	Homo
Properties	Yes	N/A	Yes	Yes	N/A	Yes	No	Yes	Yes

Algorithm	K-means [4]		
	Formalization	Heter	Homo
Properties	Yes	No	Yes

tive. In our work there are algorithms which can work in heterogeneous networks (RankClus [21]) and algorithms which are applicable in homogeneous networks (Betweenness [8]); we also include algorithms that employ objective functions to guide their clustering (Walktrap [16]) and algorithms that do not use objective functions (RankClus [21]). Some of algorithms are agglomerative (Walktrap), and some of them are divisive (Betweenness); some of algorithms can give overlapping community partition (LinkCommunity [1] and LineGraph [6]), and some of them can only give non-overlapping results (SPICi [9]).

2.2 Social Network Datasets

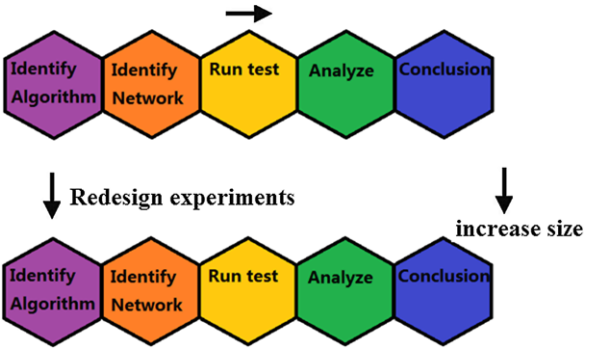
The social networks datasets are listed in Table 3. Due to the obstacle of collecting community ground-truth information, in our work we can only provide 9 datasets listed in Table 3. Our efforts are made to ensure that these datasets are representative. In order to observe the behaviors of algorithms in different sized networks, we make the networks sizes range from 34 nodes to 80,513 nodes. Besides homogeneous networks, we also include heterogeneous networks, such as Cities and Services [24].

The network datasets (or their sources) used for experimentation are: Zachary Karate Club [25], Mexican Political Power [7], Sawmill [12], Cities and Services [24], MIT Reality Mining [5], Flickr [23], Youtube [13], and LiveJournal [13]. These real world networks are selected as our datasets because they have well defined communities ground-truth information. The first five of them are small social networks and are valuable at the startup stage of our survey, by using which we can have a more intuitive and clear view of social networks and community detection algorithms. The last three are large-scale social networks, they are used to verify

Table 3 Social network datasets

Datasets	Size	Ground-truth communities	Is heterogeneous
Karate Club	34 nodes	2	No
Mexican	35 nodes	2	No
Sawmill	36 nodes	3	No
Reality Mining	79 nodes	2	No
Cities&Services	101 nodes	4	Yes
Benchmark	45 nodes	3	No
Flickr	80,513 nodes	195	No
LiveJournal	3,986 nodes	113	No
Youtube	8,202 nodes	168	No

Fig. 1 Methodology



whether the conclusions made in small social networks still hold for large-scale social networks.

2.3 Methodology

The methodology (Fig. 1) employed in our paper is something like ”black box” approach by measuring algorithms under different objective functions, because whether these objective functions are reliable or not is unknown to us, by employing this methodology we can simplify our work of experiments and achieve more concise comparisons of these performance metrics for community detection algorithms. In the first step only small size social networks with ground-truth information are chosen, which is easy for us to conclude the initial observations. With these findings we can increase the social networks size and see whether these conclusions still hold with the increment of social network size. The last three networks (Table 3) are selected to verify our conclusions when community detection algorithms work on large-scale datasets. Using this methodology we can explore more precise conclusions and unveil the relationship between network sizes and performance metrics.

3 Community Detection on Small Networks

Among our selected datasets there are heterogeneous networks and homogeneous networks, while in the set of chosen clustering algorithms there are algorithms designed for heterogeneous networks, homogeneous networks or both (Table 2). Such that if an algorithm is designed for homogeneous networks and we apply it on heterogeneous networks, it may have unreasonable results. However there is possibility that it still has high scores in some objective functions, in this way bias between objective function and ground-truth information could be identified.

3.1 Experiments on Small Networks

We firstly apply six selected community detection algorithms on the first six datasets listed in Table 3 which have small sizes. The communities detected are evaluated by performance metrics presented in Table 1.

We study these data according to two dimensions: algorithm dimension and objective function dimension. Algorithm dimension means we only study related behaviors of one specified algorithm, while objective function dimension refers that we analyze information related to specific objective function. Generally speaking the ground-truth based *rand index* is much more precise to differentiate qualities of algorithms on networks, we will compare the *rand index* scores with other metrics to unfold the bias between them.

If we focus on the RankClus algorithm we can see that metrics, such as *internal density*, *conductance*, *cut ratio* and *modularity*, to some extent can reveal the algorithm's performance over different datasets. For example RankClus has the worst performance on Mexican Political dataset when comparing with its performance on other networks in terms of *rand index*; similarly *internal density*, *modularity* and *conductance* also suggest that the detected communities are of poor quality. However there is also bias, for instance RankClus has the best performance on cities and services dataset when comparing with other algorithms in terms of *rand index*; however its related *conductance*, *cut ratio* and *modularity* are very low. Another example is, RankClus correctly clustered all nodes in the Karate Club dataset, however the *internal density* has a lower score when comparing with other algorithms.

Another interesting observation is that Walktrap algorithm and LinkCommunity algorithm have the worst performance on the same social networks (they cluster nodes of Mexican dataset and cities dataset into one single community). And more interesting thing is that while they have the worst performance (fail to partition the network) their *conductance* and *cut ratio* scores are perfect, which gives diametrically opposed evaluations. The reason for this is that Walktrap and LinkCommunity are both designed to optimizing *modularity* objective functions, Mexican dataset and cities dataset happen to have larger *modularity* than any partitions of themselves. In contrast optimization of criteria does not always lead to qualified communities. Additionally we can see that although RankClus is designed for heterogeneous networks, it also has surprisingly high scores on specific homogeneous networks. This

Table 4 Small networks experiment results

Dataset	GT	RankClus						Walktrap					
		RI	ID	C	CR	M	Co	RI	ID	C	CR	M	Co
Karate	2	**1.000**	0.275	**0.875**	**0.965**	0.211	2	0.745	0.308	0.810	0.953	0.188	2
Mexican	2	0.489	0.168	0.434	0.778	0.003	2	0.536	0.197	**1.000**	**1.000**	0.018	1
Sawmill	3	0.530	0.048	0.138	0.877	0.003	3	0.560	0.307	**0.845**	**0.981**	0.178	2
Cities	4	**0.668**	**0.988**	0.168	0.018	0.004	4	0.348	0.266	**1.000**	**1.000**	0.006	1
Reality	2	**0.575**	**1.000**	**0.987**	**0.988**	0.099	2	0.561	**1.000**	0.968	0.969	0.100	2
Bench	3	0.718	0.208	0.625	0.937	0.107	3	**1.0**	0.310	**0.874**	**0.981**	0.289	3

Dataset	GT	K-means						LinkCommunity					
		RI	ID	C	CR	M	Co	RI	ID	C	CR	M	Co
Karate	2	0.941	0.168	0.503	0.897	0.057	2	0.743	0.499	0.468	0.907	**0.284**	8
Mexican	2	0.536	0.218	0.606	0.847	0.066	2	0.536	0.197	**1.000**	**1.000**	0.018	1
Sawmill	3	0.527	0.309	0.761	0.961	0.232	3	0.560	0.328	0.731	0.902	**0.314**	5
Cities	4	0.604	0.310	0.282	0.807	**0.032**	4	0.348	0.266	**1.000**	**1.000**	0.006	1
Reality	2	0.523	0.433	0.742	0.944	**0.189**	2	0.574	0.964	0.898	0.828	0.109	3
Bench	3	**1.000**	0.143	0.411	0.888	0.015	3	0.826	0.397	0.598	0.931	**0.406**	11

Dataset	GT	SPICi						Betweenness					
		RI	ID	C	CR	M	Co	RI	ID	C	CR	M	Co
Karate	2	0.586	**0.729**	0.524	0.898	0.136	5	0.913	0.210	0.630	0.933	0.199	3
Mexican	2	0.553	**0.600**	0.648	0.903	**0.155**	3	**0.605**	0.079	0.100	0.790	0.036	7
Sawmill	3	**0.629**	**0.633**	0.547	0.947	0.192	7	0.570	0.028	0.110	0.908	0.022	6
Cities	4	0.636	0.513	0.110	0.799	0.022	12	0.267	0.000	0.000	0.729	0.007	12
Reality	2	0.573	0.88	0.844	0.900	0.098	2	0.563	0.000	0.110	0.322	0.079	9
Bench	3	0.865	**0.521**	0.731	0.965	0.260	5	0.943	0.399	0.721	0.964	0.284	4

In these tables GT states the number of classes of ground-truth, RI is the rand index score, ID is the internal density, C is the conductance, CR is the cut ratio, M represents the modularity, and Co is the number of communities detected by corresponding algorithms. As for these metrics, *higher score indicates higher quality*

is an interesting phenomenon we need to look deep into for our future work. From Table 1 we can observe that the behaviors of community detection algorithms vary in different networks.

When we concentrate on a single objective function, for instance, *internal density*, trivially we can find that SPICi algorithm has the best *internal density* on Karate dataset; however RankClus has the best performance (in terms of *rand index*) on Karate dataset. Another example is, LinkCommunity has the best *modularity* on Sawmill dataset while SPICi has the best performance (in terms of *rand index*) on Sawmill dataset. Among the data in Table 4 there are a lot of such examples, based

on current experiments results and observations, we can see that the correlation between the *rand index* and objective functions that are not based on ground-truth, is not strong.

Here we conclude our findings as below:

1. Heuristics are native reasons for behavioral differences or similarities of algorithms, similar heuristics lead to similar performance. A good example is Walktrap and LinkCommunity, although one of them generates overlapping communities while another does not, they have very similar behaviors in our selected datasets.
2. Different heuristics fit in different circumstances, inappropriate heuristics lead to damages on performance. RankClus' heuristic is applicable for most of social networks (Ranking and Clustering mutually enhance each other), however it has worst performance on the benchmark network in terms of *rand index* because the benchmark network is significantly different from social networks in most topological properties.
3. Community structure of networks depends on many factors, topological properties of networks are only parts of them. In some circumstances when topologies do not prevail, use of objective functions (highly related to topological properties) may lead to inappropriate evaluations of communities. This is the reason that *rand index* does not always agree with other metrics in our work.

3.2 Correlations Between Objective Function and Ground-Truth Measurement

Actually we can take a closer look at the bias between ground-truth measurement and objective functions by presenting their correlations quantitatively. Assume there are a set of datasets $D = \{D_1, D_2, D_3, \ldots, D_M\}$, a set of algorithms $A = \{A_1, A_2, A_3, \ldots, A_N\}$ and a set of objective functions $F = \{F_1, F_2, F_3, \ldots, F_K\}$, we apply algorithms on the given datasets and calculate corresponding objective functions scores; in this way for each pair of dataset D_i and objective function F_k there is a vector $v_{D_i,F_k} = (F_k(A_1, D_i), F_k(A_2, D_i), F_k(A_3, D_i), \ldots, F_k(A_M, D_i))$, and for each dataset there is also a vector for ground-truth $v_{D_i,G} = (G(A_1, D_i), G(A_2, D_i), G(A_3, D_i), \ldots, G(A_M, D_i))$.

By computing the correlation coefficient between v_{D_i,F_k} and $v_{D_i,G}$, we can quantitatively identify whether an objective function is a good measurement of community detection algorithm. In Fig. 2 we can observe most of objective functions on most of datasets have little correlation with ground-truth scores and some of them even have negative correlationship, such as *internal density* on karate dataset and *modularity* on reality dataset. And additionally we can see *internal density* is much more correlated with ground-truth measurement than other objective functions. From this plot we can conclude that these objective functions are not reliable enough to determine whether a community detection algorithm performs well or

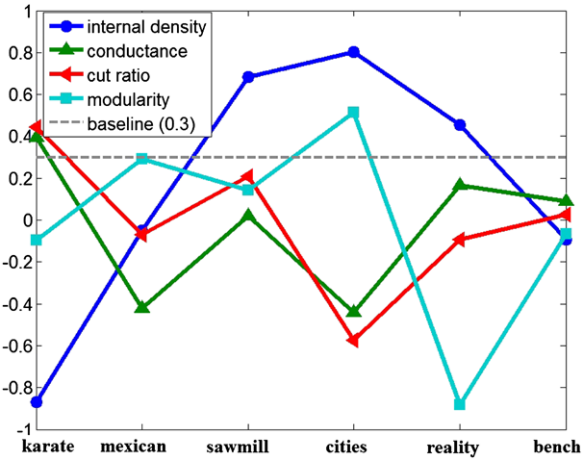


Fig. 2 Correlation coefficients between objective functions and ground-truth, small networks

not on a given network. An interesting observation is that *cut ratio* and *conductance* share the same behavior in Fig. 2. This is intuitive because both metrics are designed to capture the inter-cluster interactions.

Andrea Lancichinetti et al. proposed to use generated benchmark networks to measure the performance of community detection algorithms. However in Table 4 we can see that these six algorithms listed above all have very high *rand index* scores; however the performance on other datasets is not so promising. There are two possible reasons, the first one is that these generated benchmark networks are easy to be "mined", another one is that the generated benchmark networks are not good simulations of "real world" networks. In order to know which reason contributes to this phenomenon, we conducted more experiments and present the results in Sect. 4.1. We would like to note that, the quality of communities detected by algorithms is hard to evaluate, for example, in Fig. 3 even with ground-truth information it is still difficult for us to tell which method performs better on Karate dataset (LinkCommunity and Line Graph give almost the same *rand index* scores). Performance metrics can help our evaluations but cannot completely define the quality.

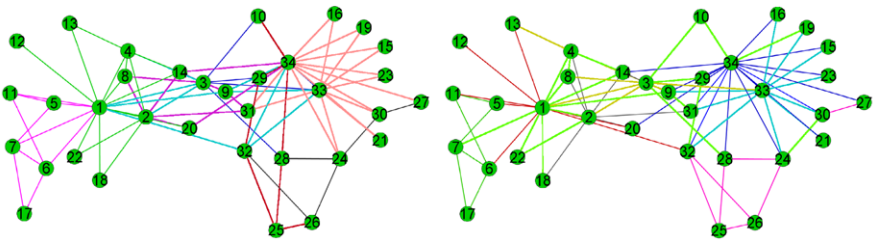


Fig. 3 LinkCommunity (*left*) and line graph (*right*) clustering on the Karate Club dataset

Table 5 Large-scale networks experiment results

Dataset	GT	RankClus						Walktrap					
		RI	ID	C	CR	M	Co	RI	ID	C	CR	M	Co
Flickr	195	0.950	**0.967**	0.108	0.998	0.079	195	0.680	0.298	**0.286**	**0.999**	**0.287**	344
Youtube	168	0.979	**0.984**	0.131	0.998	0.141	168	0.801	0.416	**0.710**	**0.999**	0.354	152
LiveJournal	113	0.977	**0.953**	0.434	0.990	0.275	114	0.981	0.487	0.780	0.990	0.365	216

Dataset	GT	K-means						LinkCommunity					
		RI	ID	C	CR	M	Co	RI	ID	C	CR	M	Co
Flickr	195	0.950	0.910	0.213	0.998	0.100	195	*	*	*	*	*	*
Youtube	168	0.979	0.931	0.396	0.990	0.128	168	0.983	0.415	0.152	0.996	**0.710**	6,701
LiveJournal	113	0.983	0.908	**0.802**	0.990	0.366	114	**0.988**	0.364	0.563	**0.999**	**0.780**	1,430

Dataset	GT	SPICi						Betweenness					
		RI	ID	C	CR	M	Co	RI	ID	C	CR	M	Co
Flickr	195	**0.960**	0.437	0.045	0.998	0.065	10,267	*	*	*	*	*	*
Youtube	168	**0.984**	0.297	0.380	0.900	0.122	816	*	*	*	*	*	*
LiveJournal	113	**0.988**	0.212	0.678	**0.999**	0.250	746	*	*	*	*	*	*

In these tables GT states the number of classes of ground-truth, RI is the rand index score, ID is the internal density, C is the conductance, CR is the cut ratio, M represents the modularity, and Co is the number of communities detected by corresponding algorithms. As for these metrics, *higher score indicates higher quality*. Results for Betweenness algorithm and partial results of LinkCommunity are not available due to their expensive computational cost and memory requirement (marked with *)

4 Community Detection on Large-Scale Networks

4.1 Experiments on Large-Scale Networks

In the above section we apply six representative community detection algorithms on six small "real world" datasets and unfold several interesting phenomenons of objective functions. In this section we increase sizes of networks and perform the same algorithms on these large-scale networks to verify whether the experimental results will be different from those on small networks.

The experimental results will also be analyzed in two dimensions. Due to computation complexity and memory requirement, results for Betweenness algorithm and part of results for LinkCommunity are not available. In Table 5 we still can see bias between ground-truth information and objective functions; for example *conductance*, *cut ratio* and *modularity* all suggest that Walktrap algorithm works better than other algorithms on Flickr dataset, however the truth is Walktrap algorithm has the lowest *rand index* score. In our observation the reliability of objective functions

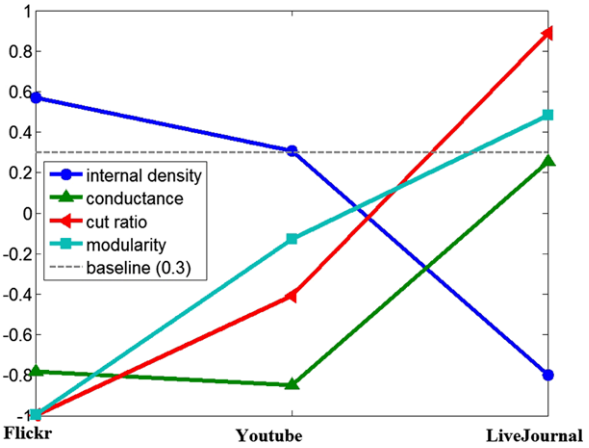


Fig. 4 Correlation coefficients between objective functions and ground-truth, large-scale networks

does not improve when the network size is increased, the bias between *rand index* and other metrics is still apparent as discussed in small size networks results. However some of them have consistent measurements with ground-truth information of some datasets, for example, *cut ratio* and *rand index* both suggest SPICi and LinkCommunity work best in LiveJournal dataset. In the same way we compute the correlation coefficients between ground-truth measurement and objective functions and plot them to verify the reliability of objective functions on large-scale networks.

In Fig. 4 we can see that our conclusions on small size networks still hold for large-scale networks, most of objective functions on most datasets have none significant correlation or even have anti-correlation. Interestingly *internal density* still performs better than other objective functions on large-scale networks.

5 Benchmark Networks

The benchmark network generator [10] can take the parameters such as, node number, average degree, and mixing parameter (Table 6) to simulate an existing "real world" network, and ground-truth information for communities is also generated. Its objective is to simulate an existing "real world" network and provide an estimated information of communities structure for this network, in this way community detection algorithms' performance can be trivially evaluated using the generated ground-truth information.

5.1 Network Model Discussion

In the work of benchmark network [10] nodes of network are partitioned into l (given by input) groups, the sizes of groups can follow some distribution specified

Table 6 Parameters for benchmark network generator [10]

Parameter	Description
n	nodes number
k	average degree
maxk	max degree
μ_t	mixing parameter
minc	minimum community size
maxc	maximum community size

in the input. Nodes of the same group are linked with a probability p_{in}, whereas nodes from different groups are connected with a probability p_{out}. Each subgraph corresponding to a group is then a random Erdős Rényi graph with connection probability $p = p_{in}$. If $p_{in} > p_{out}$ the intra-cluster density exceeds the inter-cluster density, and then the community structure of the simulated network is formed. However the Erdős Rényi graph is different from real world networks in many aspects, which are presented in Table 7.

From Table 7 we can see the Erdős Rényi network differs from real world networks in two important properties: degree distribution and clustering coefficient. The Erdős Rényi network is not a good simulation of the real world network, thus the benchmark network (based on Erdős Rényi model) discussed in [10] is unlikely to precisely simulate the real world community structures. From the network model perspective the performance of algorithms on benchmark networks cannot lead to an accurate estimation of the performance of algorithms on real world networks. We conduct several experiments in Sect. 5.2 to demonstrate the correctness of our opinion.

5.2 Experimental Results

Our experiment is to verify whether these generated networks can simulate the "real world" network precisely. For the first four datasets listed in Table 3 there are ground-truth information for communities, thus we can compare the performance of algorithm on these networks and their corresponding simulated networks to identify whether benchmark networks generator is feasible to evaluate community detection algorithms' performance. Most of the parameters requires by the benchmark network generator can be calculated trivially, such as degree, minimum community size and maximum community size; however the mixing parameter μ_t, which is set to define the proportion of each nodes links which link outside its community, is hard to computed when the community structure is not available. Leto Peel [15] proposed a novel method to estimate the mixing parameter μ_t by network structure information. In our experiments we employ Leto Peel's method to calculate the mixing parameter and use the value as the input for benchmark network generator.

Table 7 Comparison between Erdős Rényi networks and real world networks

Properties	Degree distribution	Clustering coefficient	Average diameter
Real World Networks	Power Law	High	Small
Erdős Rényi Networks	Poison	Low	Small

Table 8 Benchmark networks

Dataset	GT	RankClus		Walktrap		K-means		LinkCom		SPICi		Betweenness	
		RI	Co	RI	Co	RI	Co	RI	Co	RI	Co	RI	Co
Karate	2	1.000	2	0.745	2	0.941	2	0.743	8	0.586	5	0.913	3
Mexican	2	0.489	2	0.536	1	0.536	2	0.536	1	0.553	3	0.605	7
Sawmill	3	0.530	3	0.560	2	0.527	3	0.560	5	0.629	7	0.570	6
Reality	2	0.575	2	0.561	2	0.523	2	0.574	3	0.573	2	0.563	9

Dataset	GT	RankClus		Walktrap		K-means		LinkCom		SPICi		Betweenness	
		RI	Co	RI	Co	RI	Co	RI	Co	RI	Co	RI	Co
sim-Karate	2	0.510	2	0.520	4	0.510	2	0.520	28	0.500	6	0.510	3
sim-Mexican	2	0.510	2	0.510	8	0.510	2	0.500	1	0.510	3	0.500	6
sim-Sawmill	3	0.630	3	0.610	7	0.610	3	0.630	1	0.630	9	0.630	2
sim-Reality	2	0.490	2	0.500	2	0.490	2	0.500	1	0.500	2	0.490	2

In these tables GT states the number of classes of ground-truth, RI is the rand index score, and Co is the number of communities detected by corresponding algorithms. As for these metrics, *higher score indicates higher quality*

By the information listed in Table 8 we simulate 100 networks for first four datasets listed in Table 3, cities and services dataset can not be simulated because benchmark network generator is not able to simulate heterogeneous network. We apply selected algorithms on these 100 networks for each dataset, compute the rand index scores and then calculate the average rand index score for each algorithm on 100 simulated networks. The results are listed in Table 8. The top phase of the table shows the rand index scores of each algorithm on each dataset, the bottom phase presents the performance of each algorithm on each dataset's simulated network. Trivially we can see the "real world" dataset can easily differentiate algorithms based on their performance, for example, RankClus outperforms others on karate dataset and SPICi performs much better than others on sawmill dataset, while for the simulated networks algorithms almost have the same scores on the same dataset. In conclusion, 1) it is hard to differentiate the performance of community detection algorithms on benchmark networks; 2) the behaviors of community detection algorithms on real world networks are different from their behaviors on corresponding benchmark networks; 3) benchmark networks are not promising substitutes of real world datasets for algorithms measurements.

6 Conclusion

Seven representative algorithms were compared under various performance metrics, and on various "real world" social networks, from small size networks to large-scale networks. Based on our current observations of experiments results, we can conclude that performance metrics based on the ground-truth information are more reliable than objective functions that are not based on ground-truth, such as *internal density* and *modularity*. And the reliability of non ground-truth based objective functions does not improve with the increment of network size. Characteristics of different algorithms are unfold in our experiments, RankClus has best or comparable performance on most datasets due to the reason that it employs a more general heuristic instead of using objective functions to guide clustering process.

In our work we also discuss the benchmark networks with built-in communities structures. We analyzed the differences between benchmark networks and real world networks, such as degree distribution and clustering coefficient. These differences lead to the invulnerability of benchmark networks as they are used to measure the performance of community detection algorithms designed for real world networks. By experiments we conclude that the networks created by the benchmark network generator [10] are not qualified enough to differentiate the performance of community detection algorithms; algorithms tend to have similar scores in given simulated networks.

7 Future Work

Our current work has included the experiments on small networks, large-scale networks and benchmark networks and draw several conclusions. For example objective functions not based on ground-truth information are not reliable to accurately reveal the performance of algorithms on social networks we have studied. In the future work more performance metrics are to be involved, and more algorithms and datasets will be selected to reinforce the robustness of our conclusions. With the gradual increment of dataset size the relation between social network volume and objective functions is estimated to be unfolded. Next, the networks studied will be expanded into other genres than social networks, such as biological networks and telecommunication networks, and algorithms and performance metrics will be compared independently in each category of networks. Much more objective functions will be included into our future study, which is designed to conduct an empirical comparison of algorithms for network community detection. In this way we can expand our work into other areas than social networks, the different behaviors of objective functions in different types of networks may be unfold in such experiments.

Acknowledgements This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or

References

1. Ahn Y, Bagrow JP, Lehmann S (2010) Link communities reveal multiscale complexity in networks. arXiv:0903.3178v3 [physics.soc-ph]
2. Chen J, Zaïane OR, Goebel R (2009) Detecting communities in social networks using max-min modularity. In: International conference on data mining (SDM 09)
3. de Nooy W, Mrvar A, Batagelj V (2004) Exploratory social network analysis with Pajek, Chapter 12. Cambridge University Press, Cambridge
4. Dhillon I, Guan Y, Kulis B (2005) A fast kernel-based multilevel algorithm for graph clustering. In: Proceedings of the 11th ACM SIGKDD, Chicago, IL, August 21–24
5. Eagle N, Pentland A (2006) Reality mining: sensing complex social systems. Pers Ubiquitous Comput 10(4):255–268
6. Evans TS, Lambiotte R (2009) Line graphs, link partitions, and overlapping communities. Phys Rev E 80(1):016105
7. Gil-Mendieta J, Schmidt S (1996) The political network in Mexico. Soc Netw 18(4): 355–381
8. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci USA 99(12):7821–7826
9. Jiang P, Singh M (2010) SPICi: a fast clustering algorithm for large biological networks. Bioinformatics 26(8):1105–1111
10. Lancichinetti A, Fortunato S, Kertész J (2009) Detecting the overlapping and hierarchical community structure in complex networks. New J Phys 11(3):033015
11. Leskovec J, Lang KJ, Mahoney MW (2010) Empirical comparison of algorithms for network community detection. In: WWW 2010, April 26–30, Raleigh, North Carolina, USA
12. Michael JH, Massey JG (1997) Modeling the communication network in a sawmill. For Prod J 47:25–30
13. Mislove A (2009) Online social networks: measurement, analysis, and applications to distributed information systems. Ph.D Thesis, Rice University, Department of Computer Science
14. Pandit S, Kawadia V, Yang Y, Chawla NV, Sreenivasan S (2011) Detecting communities in time-evolving proximity networks. In: IEEE first international workshop on network science (submitted)
15. Peel L (2010) Estimating network parameters for selecting community detection algorithms. In: 13th international conference on information fusion
16. Pons P, Latapy M (2006) Computing communities in large networks using random walks. J Graph Algorithms Appl 10(2):191–218
17. Radicchi F, Castellano C, Cecconi F, Loreto V, Parisi D (2004) Defining and identifying communities in networks. Proc Natl Acad Sci USA 101(9):2658–2663
18. Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905
19. Steinhaeuser K, Chawla NV (2010) Identifying and evaluating community structure in complex networks. Pattern Recognit Lett 31(5):413–421
20. Steinhaeuser K, Chawla NV Is modularity the answer to evaluating community structure in networks? In: International conference on network science (NetSci), Norwich, UK
21. Sun Y, Han J, Zhao P, Yin Z, Cheng H, Wu T RankClus: integrating clustering with ranking for heterogeneous information network analysis. In: EDBT 2009, March 24–26, 2009, Saint Petersburg, Russia
22. Sun Y, Han J (2010) Integrating clustering and ranking for heterogeneous information network analysis. In: Yu PS, Han J, Faloutsos C (eds) Link mining: models, algorithms and applications. Springer, New York, pp 439–474

23. Tang L, Liu H (2009) Scalable learning of collective behavior based on sparse social dimensions. In: Proceedings of the 18th ACM conference on information and knowledge management (CIKM'09)
24. World Cities and Global Firms dataset was created by Taylor PJ, Walker DRF as part of their project "World city network: data matrix construction and analysis" and is based on primary data collected by Beaverstock JV, Smith RG, Taylor PJ (ESRC project "The geographical scope of London as a world city" (R000222050))
25. Zachary WW (1977) An information flow model for conflict and fission in small groups. J Anthropol Res 33:452–473

A Study of Malware Propagation via Online Social Networking

Mohammad Reza Faghani and Uyen Trang Nguyen

Abstract The popularity of online social networks (OSNs) have attracted malware creators who would use OSNs as a platform to propagate automated worms from one user's computer to another's. On the other hand, the topic of malware propagation in OSNs has only been investigated recently. In this chapter, we discuss recent advances on the topic of malware propagation by way of online social networking. In particular, we present three malware propagation techniques in OSNs, namely cross site scripting (XSS), Trojan and clickjacking types, and their characteristics via analytical models and simulations.

Keywords Malware propagation · Online social networks · XSS worm

1 Introduction

Online social networks (OSNs) such as Facebook, Twitter and MySpace have provided hundreds of millions of people worldwide with a means to connect and communicate with their friends, family and colleagues geographically distributed all around the world. The popularity and wide spread usage of OSNs have also attracted attackers and hackers who would use OSNs as a platform to propagate automated worms from one user's computer to another's.

The population of potential victims of web-based malware is much larger than that of other types of worms due to the popularity of the world wide web. In addition, web-based worms are not banned through web proxies and network address translation (NAT) processes. Research has shown that web-based malware can propagate much faster than traditional malware [13]. As a matter of fact, the first OSN worm that hit MySpace in 2005 by exploiting a cross site scripting vulnerability in a MySpace web application infected about one million victims within 24 hours [13].

M.R. Faghani (✉) · U.T. Nguyen
York University, Department of Computer Science and Engineering, Toronto, ON M3J 1P3, Canada
e-mail: faghani@cse.yorku.ca

U.T. Nguyen
e-mail: utn@cse.yorku.ca

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_13,

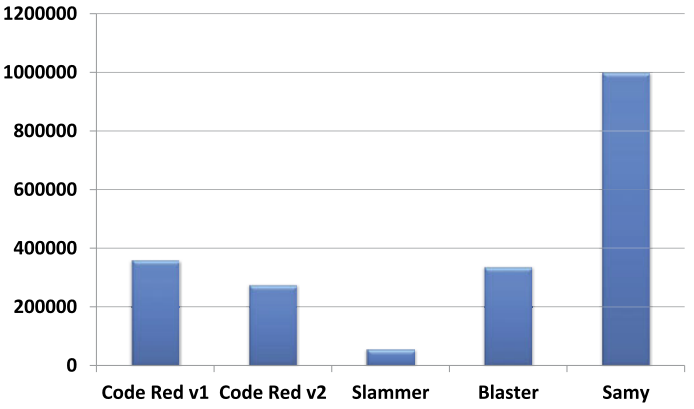


Fig. 1 Total number of infections caused by different computer worms in 20 hours

As Fig. 1 shows, this worm, which was named Samy, propagated much faster other traditional computer worms.

The topic of malware propagation in OSNs has only been investigated recently [8, 10, 11, 23, 28, 29]. The objective of this chapter is to discuss recent advances on this topic. In particular, we present three malware propagation techniques in OSNs, namely XSS, Trojan and clickjacking types, and their characteristics via analytical models and simulations.

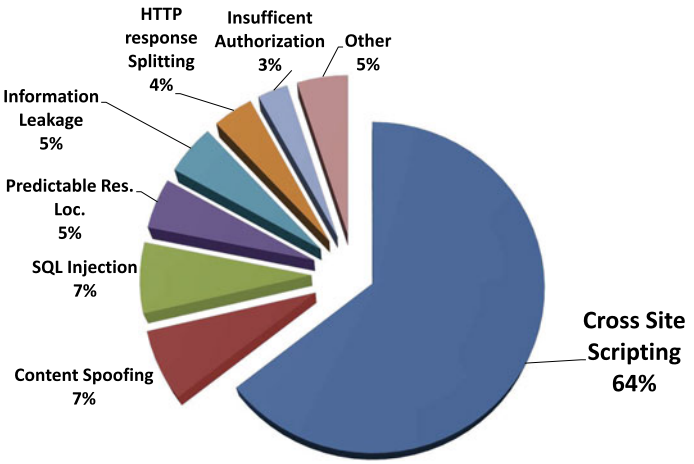
The remainder of this chapter is organized as follows. In Sect. 2, we briefly describe three types of malware propagating via online social networking. In Sect. 3, we discuss the characteristics of OSNs and algorithms used for generating simulated OSN graphs. A simulation-based study of malware propagation in OSNs is presented in Sect. 4. In Sect. 5, we review analytical models characterizing the propagation of malware in OSNs. We discuss OSN malware countermeasures in Sect. 6 and related work in Sect. 7. We then summarize the chapter in Sect. 8.

2 Different Types of Malware

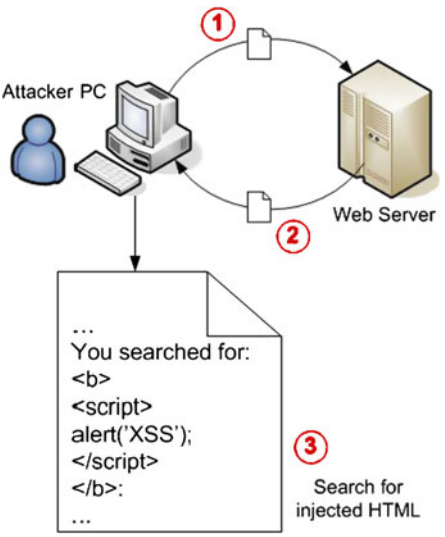
In this section, we briefly discuss the main characteristics of different types of malware propagating through OSNs. There exist currently three different types of OSN malware: cross site scripting, Trojan and clickjacking worms.

2.1 Cross Site Scripting Worms

Cross site scripting (XSS) is a security flaw to which many web applications are vulnerable [13, 20]. The graph in Fig. 2(a) shows the distribution of web application vulnerabilities, among which XSS is the most common threat. While XSS is a



(a) Distribution of Vulnerabilities in web applications



(b) Reflective XSS attack [13]

Fig. 2 Cross site scripting can be used to create self propagating worms

common vulnerability in web applications, its threat becomes more noticeable due to the combination of HTML and Asynchronous JavaScript and XML (AJAX) technologies. AJAX allows a browser to issue HTTP requests on behalf of the user. Thus there is no need for an attacker to trick the user into clicking a malicious link. This technique provides facility for malware writers to create XSS worms.

There are two types of XSS attacks: persistent and non-persistent [20]. In persistent attacks (also known as stored attacks), the injected code is permanently stored on a target server as HTML text in a database, a comment field, or messages posted on online forums. A victim's computer then accesses the malicious code on the server when it retrieves the stored information via the web browser. Non-persistent attacks (also known as reflective attacks) are the more common type of XSS attacks. In this case, the injected code is sent back to the visitor by the server in an error message, a search result, or any other type of response that reflects some or all of the user's input in the result (Fig. 2(b)). Since the reflected response contains the malicious code, the visitor's browser will interpret the code, execute the malware, and infect his/her computer.

An XSS worm, also known as a cross site scripting virus, is a malicious code that propagates itself automatically among visitors of a website in an attempt to progressively infect other visitors. XSS worms can be considered as a hybrid of stored and reflected XSS attack. This type of worms has the ability to copy itself from a source to another part of the Internet using existing XSS vulnerabilities of web applications.

An XSS worms infect members of a social network in two steps. The worm creator first adds the malicious payload to his profile, e.g., in the form of a link. Subsequently, any person who visits this profile will get infected and the malicious payload will be added to the visitor's profile due to an AJAX technique and an XSS flaw. The visitor's profile then becomes an infectious profile, which allows the worm to propagate as a new infection source [10, 11].

2.2 Trojan Malware

The best known OSN Trojan worm is Koobface [15], which was first detected in 2008. It spread itself in both MySpace and Facebook by sending messages with interesting topics using social engineering techniques to deceive people into opening the messages. Such a message directed the recipients to a third-party website unaffiliated with Facebook where they were prompted to download what was claimed to be an update of the Flash player. If they downloaded and executed the file, they would infect their computers with Koobface. The infected machine turned into a zombie or a bot. Moreover, the owner of the infected profile unknowingly sent out messages to all people on his/her friend list, allowing the worm to propagate further in the network.

2.3 Clickjacking Malware

Clickjacking worms are also known as "likejacking" or "user interface (UI) redressing". A clickjacking attacker creates a website that shows a counterfeit YouTube

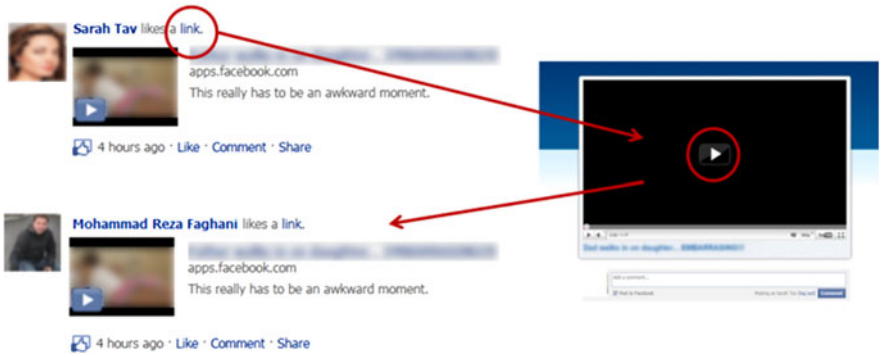


Fig. 3 Clickjacking technique used to spread spam messages that may lead to malicious software

video player, or other graphical icons, and invites the victim to click on a play button to view the video. What really happens is that the victim is clicking a Facebook "Like" button that has been hidden beneath the images using a method of coding called UI redressing. What the victim has just "Liked" is then displayed on his wall, which in turn may attract his/her friends to click on that link and become new sources of infection (Fig. 3). Since Trojan and clickjacking worms operate and propagate in a similar manner, we consider them as one type in this chapter.

Clickjacking worms could be combined with Trojan malware to create a new hybrid type but, to the best of our knowledge, no such malware has been created or deployed yet.

3 Characteristics of Online Social Networks

An OSN can be represented by an equivalent graph in which each vertex (or node) represents a person and a link between two vertices indicates the existence of a relationship between the two respective persons. To simplify the discussions in this chapter, we generalize relationships between OSN users as friendship. (In some OSNs such as LinkedIn, relationships can be colleagues or business contacts). Studies have shown that real-world social networks are highly clustered small-world networks [26] with a degree distribution often following a power law distribution. The characteristics of *online* social networks can be summarized as follows [7, 14, 30]:

1. An OSN typically has a low average network distance, approximately equal to $\log(s)/\log(d)$, where s is the number of vertices (people), and d is the average vertex degree of the equivalent graph.
2. Online social networks typically show a high clustering property, or high local transitivity. That is, if person A knows B and C, then B and C are likely to know each other. Thus A, B and C form a friendship triangle. Let k denote the degree

of a vertex v. Then the number of all possible triangles originated from vertex v is $k(k-1)/2$. Let f denote the number of friendship triangles of a vertex v in a social network graph. Then the clustering coefficient $C(v)$ of vertex v is defined as $C(v) = 2f/(k(k-1))$. The clustering coefficient of a graph is the average of the clustering coefficients of all of its vertices. In a real OSN, the average clustering coefficient is about 0.1 to 0.7.

3. Node degrees of a social network graph tend to be, or at least approximately, power-law distributed. The node degree of a power-law topology is a right-skewed distribution with a power-law Complementary Cumulative Density Function (CCDF) of $F(k) \propto k^{-\alpha}$, which is linear on a logarithmic scale. The power law distribution states that the probability for a node v to have a degree k is $P(k) \propto k^{-\alpha}$, where α is the power-law exponent.

There exist few algorithms that can generate social network graphs with the above characteristics [6, 7, 14]. For the simulations reported in this chapter, we used the algorithm proposed by Holme and Beom [14], due to the fact that it can be fine tuned to generate a social network graph with a desired clustering coefficient and power law distribution of node degrees.

In one of our experiments, we evaluated the speed of malware propagation as a function of clustering coefficients. For this experiment, we would need to vary the clustering coefficient while keeping other parameters of the network graph such as the maximum and average node degrees constant. To create such similar graphs with different clustering coefficients, we would need random graph generation algorithms such as random rewiring or the algorithm by Viger and Latapy [25]. In this section, we discuss the algorithm by Holme and Beom along with these random graph generation algorithms. We call a random graph generated based on an OSN graph an *equivalent random graph* (ERG).

3.1 Holme's Social Network Graph Generation Algorithm

This algorithm is based on the algorithm proposed by Barabasi and Albert [2], which we term the BA algorithm. The objective of the BA algorithm is to create graphs with node degrees following power law distributions. These graphs have short average network distances typical of OSNs, but they may not have high clustering coefficients, between 0.1 to 0.7, to faithfully model social network graphs [14]. This motivated Holme and Beom to modify the BA algorithm to generate graphs having high clustering coefficients typical of OSNs.

The BA algorithm works as follows:

1. The initial condition: A graph consists of m_0 vertices and no edges.
2. The growth step: One new vertex v with m edges is added to the above graph at every time step. Time t is identified as the number of time steps.

Table 1 Parameters of the simulated OSN and its ERG

Parameter	Graphs	
	OSN graph (Holme and Beom)	ERG (Viger and Latapy)
	Value	Value
Number of vertex (people)	10000	10000
Number of edges	29990	29990
Average clustering coefficient	0.14	0.0035
Average shortest path length	5.13	4.4
Network diameter	10	8
Maximum node degree	190	190
Average node degree d	5.99	5.99
$\log(n)/\log(d)$	5.14	5.14

3. The preferential attachment (PA) step: Each of the m edges incident on v is then attached to an existing vertex u with the probability P_u defined as follows:

$$\frac{k_u}{\sum_{i \in v} k_i} \tag{1}$$

In the above equation, k_i represents the degree of node i, and V is the set of vertices of the current graph. The growth step is iterated N times, where N is the total number of vertices (users) in the final OSN graph. Every time a vertex v with m edges is added to the network, the PA step is performed m times, once for each of the m edges incident on v. After t time steps, the BA network graph will contain $m_0 + t$ vertices.

To increase the clustering coefficient, Holme and Beom suggested a new step called triad formation (TF). If, in a PA step, an edge between u and v is formed, then a TF step will attempt to add another edge between v and an arbitrary neighbor w of u. If all neighbors of u have already been connected to v, the TF step is skipped and a new PA step will start.

In each iteration, a PA step is first performed: a vertex v with m edges is added to the existing network. Then a TF step is executed with probability P_t. The average number of the TF trials per added vertex is given by $m_t = (m - 1) \times P_t$ which is a control parameter in Holme's algorithm. It has been shown that the degree distribution of any graph generated by Holme's algorithm will have node degrees following a power law distribution with $\alpha = 3$.

We used Holme's algorithm to generate a graph that has the characteristics of a social network and the following parameters: $\alpha = 3$, $N = 10000$, $m_0 = 3$, $m = 3$ and $m_t = 1.8$. The parameters of the resulting social network graph are listed in Table 1.

As Table 1 shows, the synthesized OSN graph satisfies all the three required characteristics of an OSN. The average shortest path length of the graph is 5.13, which is less than $\frac{\log n}{\log d} = 5.14$. The clustering coefficient is moderate, approximately 0.14. The degrees of the vertices follow a power law distribution, as shown in Fig. 4.

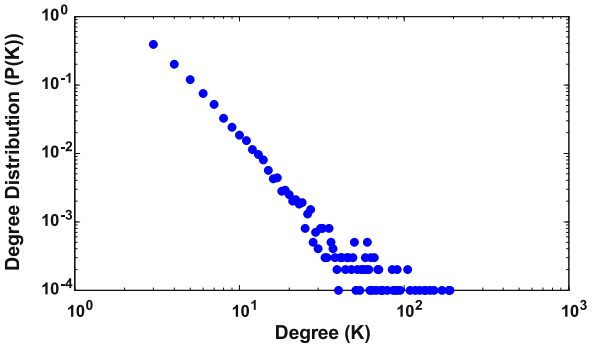


Fig. 4 The degrees of the vertices of the resulting graph follow a power law distribution

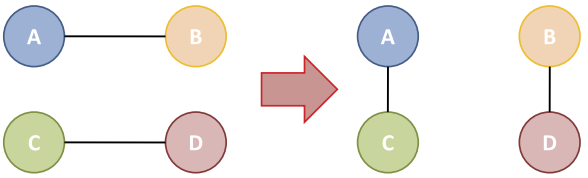


Fig. 5 Random rewiring technique to generate equivalent random graphs

3.2 Random Graph Generation Algorithms

To evaluate the impacts of clustering coefficients on worm propagation, we would need to keep the other parameters of a graph such as the maximum and average node degrees constant while varying the clustering coefficient. Such graphs with different clustering coefficients can be generated as equivalent random graphs (ERG). Given a network graph, an ERG with the same node degree distribution can be generated using random rewiring or the algorithm proposed by Viger and Latapy [25].

In the random rewiring scheme, we randomly select a pair of edges and "substitute" the edges as shown in Fig. 5. The random selection and substitution are done until we obtain the desired clustering coefficient.

The random rewiring algorithm generates ERGs that have strong correlations to the original graph. We would want random graphs that are as versatile as possible while maintaining the same node degree distribution of the original graph. Therefore, we chose the algorithm proposed by Viger and Latapy [25] to generate equivalent random graphs for our simulations. Such a random graph has the same degree distribution as the original network graph, but different characteristics such as a different clustering coefficient, average shortest path length or network diameter. The parameters of the OSN created earlier and its ERG are listed in Table 1.

Following are some observations obtained from comparing the ERG and the corresponding OSN graph (Table 1). The average clustering coefficient of the original OSN graph is about 40 times higher than that of the similar random graph, 0.14 vs. 0.0035. This reflects the high clustering characteristic of OSNs. Also, the average

shortest path length of the original OSN graph is longer than that of the ERG, 5.13 vs. 4.4. The network diameter of the OSN is 10 hops compared to eight hops in the ERG. These differences are due to the small-world phenomenon described by Watts [26].

4 Simulation of Malware Propagation in Online Social Networks

In this section, we present our simulation results on malware propagation in OSNs. The simulation software is implemented in MATLAB. The simulation is of discrete-event type, consisting of discrete time slots. In each time slot, a user (node) is chosen randomly and the user will perform an action such as visiting a profile (for XSS malware), or executing the malware (for Trojan/clickjacking malware). In all the simulations presented below, we use the synthesized OSN created using Holme's algorithm [14] and its ERG created using the algorithm by Viger and Latapy [25] whose parameters are listed in Table 1, unless otherwise stated. The performance metric is the total number of infected profiles (users) over time, assuming an initial number of infected profiles of one, unless otherwise stated. Each data point in the result graphs is the average of 100 runs, each with a different random seed. We measure the total number of infected profiles over time as functions of the following parameters:

- *Visiting-friends probability* q. The probability that a user visits his/her friends' profiles versus strangers' profiles. A friendship exists between two users u and v if there is an edge connecting nodes u and v in the network graph.
- *Graph structure*. A graph can be an original social network graph created using Holme's algorithm, or an equivalent random graph generated based on an original social network graph using the algorithm by Viger and Latapy [25].
- *Probability p of executing the malware by a user*. For Trojan or clickjacking worms, this is the probability that a user will click on a malicious link and execute the malware.
- *Node degree threshold for visiting friends vs. strangers*. Let M be the maximum node degree in the network, and K_c be a threshold factor ($0 \leq K_c \leq 1$). Nodes with degrees less (more) than the threshold $K_c M$ will visit their friends less (more) frequently than strangers. That is, users whose numbers of friends (node degrees) are lower than the threshold $K_c M$ will visit their friends with probability $q \leq 0.5$ and visit strangers with probability $1 - q$.
- *Initial number of infected profiles* (*users*).

We first present the simulation results on XSS worm propagation, followed by the results on Trojan and clickjacking worms. As mentioned earlier, Trojan and clickjacking worms spread in a similar fashion since both send the malware globally to all friends of a user. They are thus treated as one type in our simulations.

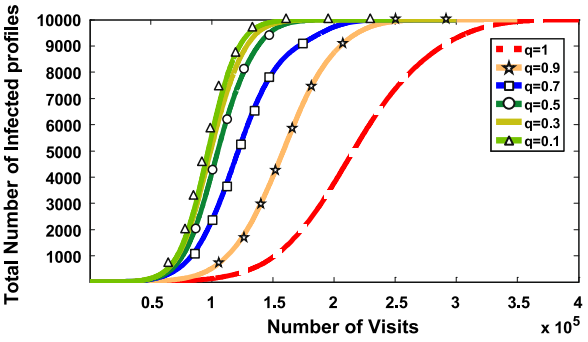


Fig. 6 XSS worm propagation for different values of probability q

4.1 Simulation of XSS Worm Propagation

In order for an XSS worm to propagate, a vulnerable user has to visit an infectious profile (user) to get infected. The user's vulnerability is determined by whether or not the user's web browser is able to execute the malicious script. (A browser may not be able to execute the script because there is an Anti-Virus active or the user has disabled JavaScript using special add-ons such as NoScript for a Firefox browser.)

In the simulation of XSS worm propagation, an event is defined as a single visit to the social network website. If the visitor is vulnerable and visits an infected profile, then the visitor will get infected.

4.1.1 Effects of Visiting-Friends Probability q

In each time unit, an uninfected user is chosen randomly using a uniform distribution. The user then visits one of his/her friends with probability q, or picks randomly a stranger to visit with probability of $1 - q$. We assume that all users have the same visiting-friends probability q. Figure 6 shows the trend of XSS worm propagation for different values of q from 0.1 to 1. The simulation results show that if people visit their friends more often than strangers, the propagation speed will be slower. This confirms the analytical model proposed by Faghani and Saidi in [11], which will be described in Sect. 5.2.

4.1.2 Effects of the Network Graph Structure

In this experiment, we examine the effects of the clustering coefficient on the speed of XSS worm propagation using simulations. We use the social network graph and its ERG whose parameters are listed in Table 1. We assume a visiting-friends probability $q = 0.9$ on both network graphs. Figure 7 shows the trends of XSS worm propagation for both graphs. Although both networks share the same visiting-friends probability and other parameters (e.g., maximum and average node degrees), their results are different. The propagation is slower in the original OSN graph (i.e., the

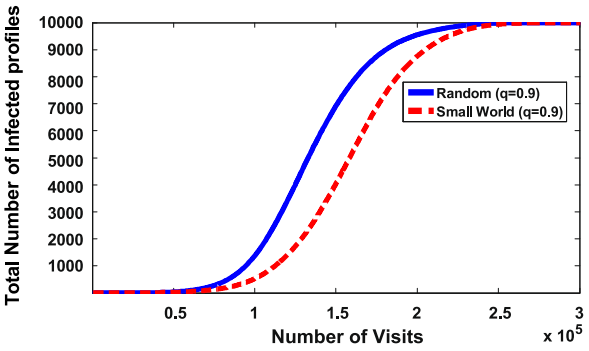


Fig. 7 XSS worm propagation in the OSN vs. its ERG with $q = 0.9$

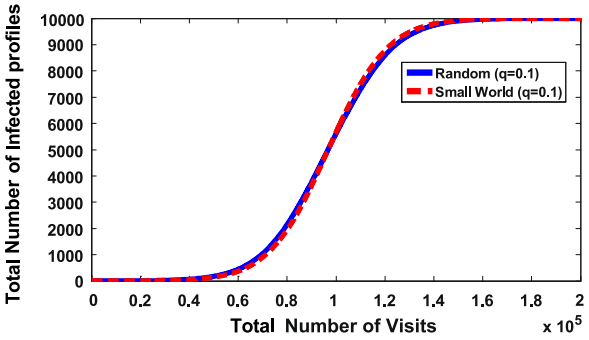


Fig. 8 XSS worm propagation in the OSN vs. its ERG with $q = 0.1$

small-world graph) thanks to its highly clustered structure, which makes the malware circulate within a cluster for a while before making its way to other parts of the network. In short, the high clustering characteristic of OSNs helps slow down the propagation of malware. (However, this fact has not been considered in existing analytical models of malware propagation in OSNs.)

We repeated the above experiment, but changed the visiting-friends probability to $q = 0.1$. The results in Fig. 8 show that both networks experienced the same XSS worm propagation speed in this case. This implies that the network topology is meaningful only with high probabilities of visiting friends. A low visiting-friends probability means that a malware is distributed from one community to another in the network more often than being contained within that community. Therefore, the highly clustered structure of the OSN does not help in this case, and the XSS worm propagation speed is similar to that in the ERG network.

4.1.3 Effects of Node Degree Threshold for Visiting Friends vs. Strangers

Initially when users join an online social network, they start looking for friends by visiting different profiles. Hence, they visit strangers more frequently than their friends. After establishing friendships with a set of people, a user tends to socialize with his/her friends and thus visit their profiles more often than strangers'.

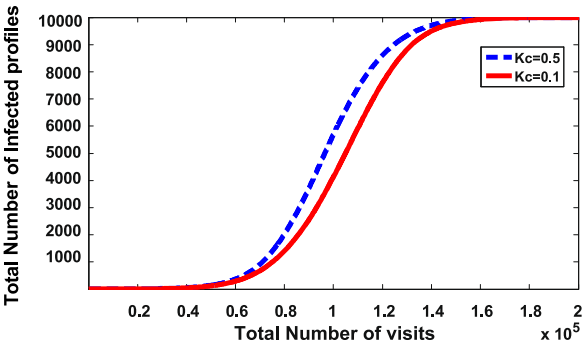


Fig. 9 XSS worm propagation given $K_c = 0.1$ and $K_c = 0.5$

In this simulation, we assume that a fraction of people with low numbers of connections will visit strangers more frequently than their friends. Let M be the maximum node degree in the network, and K_c be a threshold factor ($0 \leq K_c \leq 1$). Nodes with degrees less (more) than the threshold $K_c M$ will visit their friends less (more) frequently than strangers. In this experiment, users whose numbers of friends (node degrees) are lower than the threshold $K_c M$ will visit their friends with probability q_1, where q_1 has a normal distribution with $\mu = 0.25$ and $\sigma = 0.05$. The other users (i.e., those whose node degrees are higher than the threshold $K_c M$) will visit their friends with probability q_2, where q_2 has a normal distribution with $\mu = 0.75$ and $\sigma = 0.05$. The trends of worm propagation for are $K_c = 0.1$ and $K_c = 0.5$ (equivalent to a threshold of 19 friends and 95 friends, respectively, given the network in Table 1 which has a maximum node degree of 190) are shown in Fig. 9.

The results show that the propagation is slower when $K_c = 0.1$. The reason is that when $K_c = 0.1$ there are 281 users that have more than 19 friends in contrast to 12 users having more than 95 friends when $K_c = 0.5$. When $K_c = 0.1$, more users visit their friends more frequently than when $K_c = 0.5$, which leads to slower worm propagation.

4.2 Simulation of Trojan and Clickjacking Malware Propagation

In each time step, a user is randomly selected, who will check his/her messages sent via the OSN messaging system. This action is also called a *visit* or an *event*. With probability p, the user follows the malicious link in the message and executes the malware. Once the user gets infected, the Trojan code sends a similar message containing the malicious link to all of the user's friends (or posts a message on his/her wall as done by a Facebook clickjacking worm).

4.2.1 Effects of the Probability p of Executing the Malware by a User

The trend of Trojan malware propagation is shown in Fig. 10 for different distributions of probability p listed in Table 2. The graphs demonstrate that the higher the

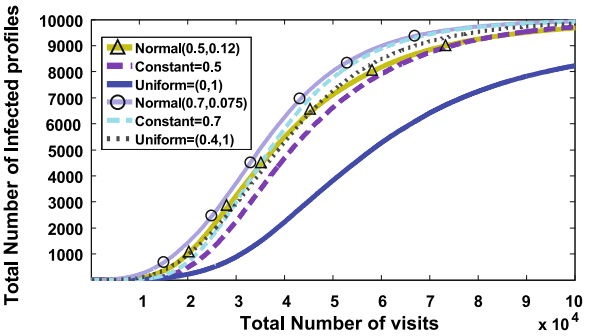


Fig. 10 Trojan worm propagation given different distributions of p

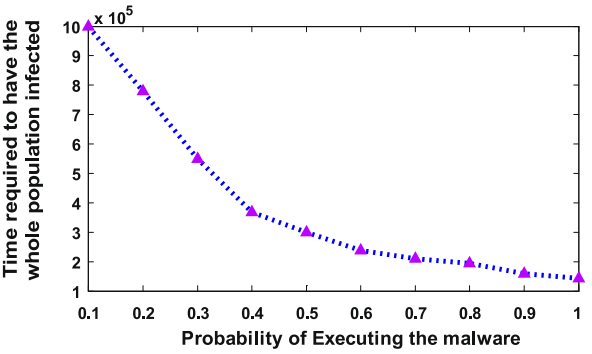


Fig. 11 Propagation speed increases exponentially with probability p [8]

Table 2 Probabilities p in Trojan worm propagation

Distribution	Value
Constant	$p = 0.5$
	$p = 0.7$
Normal distribution	$\mu = 0.5$ and $\sigma = 0.12$
	$\mu = 0.7$ and $\sigma = 0.075$
Uniform distribution	Range [0, 1] with $\mu = 0.5$
	Range [0.4, 1] with $\mu = 0.7$

probability of executing the malware, the faster the worm propagates. Given p with a uniform distribution having $\mu = 0.5$ from $p = 0$ to $p = 1$, the propagation is very slow because very few people open the messages and follow the malicious links.

Faghani et al. [8] show that increasing the value of p will speed up the propagation *exponentially*. The authors used a synthesized OSN with 10000 nodes, 29991 edges, and a clustering coefficient of 0.16. They then measured the number of visits (events) needed to get all the users infected as a function of probability p, starting with one infected user. Their result, shown in Fig. 11, indicates that the propagation

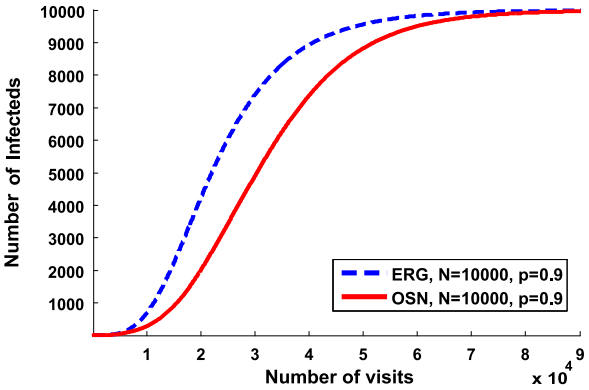


Fig. 12 Trojan worm propagation in the OSN vs. its ERG with $p = 0.9$

speed increases exponentially with higher probability of p. The result highlights the importance of raising awareness of malware threats among OSN users.

4.2.2 Effects of the Clustering Coefficient on Trojan Propagation

Like XSS worm propagation, Trojan worm propagation is also affected by the highly clustered structure of OSNs. We examine this effect using the OSN and its ERG listed in Table 1 and assuming a malware execution probability $p = 0.9$. The result in Fig. 12 show that the malware propagates more slowly in the OSN than in the ERG network. This is consistent with the observation from the XSS malware experiment presented in Sect. 4.1.2.

4.2.3 Trojan vs. XSS Propagation

In this experiment, we compare the propagation speed of Trojan and XSS worms under the same network conditions and parameters. We assume that people visit other users (profiles) following a Poisson process with an average of k times per minute. Thus the interval between visits follows an exponential distribution with an average of $\frac{1}{k}$. For the XSS malware, we assume a visiting-friends probability $q = 0.9$. For the Trojan (Koobface) worm, we consider two malware executing probabilities of $p = 0.5$ and $p = 0.7$. Given $k = 10$, Fig. 13 shows the results of propagation speeds of Trojan and XSS worms in the 10000-node OSN defined in Table 1. The results demonstrate that the propagation speed of Trojan worms is faster than that of XSS worm in OSNs.

To explain the above result intuitively, we consider a tiny social network having five users as depicted in Fig. 14. Initially user A is infected while the others are not. If user A is infected with a Trojan malware and she has already sent the malicious message to all her friends, B, C, D and E, then in the next visit (event) an uninfected user will be selected with a probability of $\frac{4}{5}$. Assuming that this user will open the

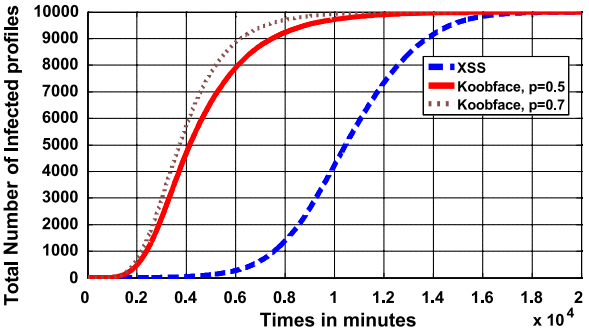


Fig. 13 Propagation speed of Trojan vs. XSS malware

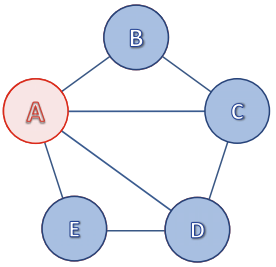


Fig. 14 A tiny social network in which user A is initially infected while the others are not

message and follow the malicious link, this means that one of the uninfected users will get infected with probability of $\frac{4}{5}$.

However, if user A is infected with an XSS worm, then in the next visit, an uninfected user will be selected with a probability of $\frac{4}{5}$ and this user will visit the infected user A with a probability of $\frac{1}{4}$. Therefore, one of the uninfected users will get infected with probability of $\frac{4}{5} \times \frac{1}{4} = \frac{1}{5}$. This explains why the Trojan worm propagated much faster than the XSS worm in the above simulation.

In other words, Trojan worms are more proactive than XSS worms. They present themselves to users (in the form of messages) so as to be activated and propagated, while an XSS worm sits on an infected profile waiting for users to select and visit the profile.

4.2.4 Effects of the Initial Number of Infected Profiles

Another important parameter that should be considered in the propagation speed of Trojan and XSS worms is the initial number of infected profiles (users) i_0. Using the OSN graph listed in Table 1, we varied the initial number of infected profiles from 50 to 500, and measured the number of visits (events) required to get 90 % of the population (i.e., 9000 users) infected. The obtained results were then normalized to the maximum number of visits measured for each type of worm. Figure 15 shows the results of this experiment. We can see that the effect of increasing the initial number of infected profiles for a Trojan worm such as Koobface is not noticeable.

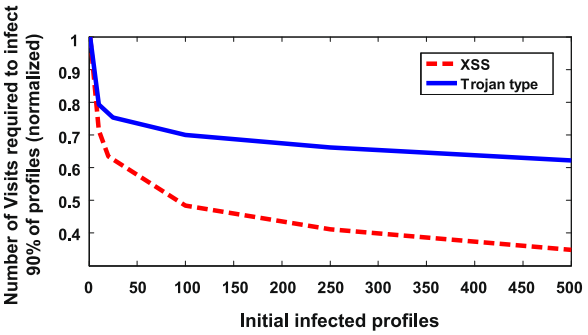


Fig. 15 XSS versus Trojan worm propagation for different values of the number of initial infected profiles

However, increasing the initial number of infected profiles for XSS leads to faster propagation.

In Trojan malware propagation, assume that each person follows the malicious link and executes the malware code with a probability of 1. Suppose that there are initially n infected profiles. Thus, on average, there are $n \times \overline{\deg(n)}$ potential targets for infection, where $\overline{\deg(n)}$ is the average degree of the n infected nodes. Therefore by choosing a large value for the initial number of profiles infected by a Trojan malware i_0, it is possible to make almost all members become potential targets for infection. Thus if we keep increasing i_0 after that, the increase does not have significant effects on the propagation speed.

XSS malware, in contrast, requires a user to visit an infected profile to get infected. Therefore, if we increase the initial number of infected profiles, we practically speed up the propagation of the malware in the network.

In Sect. 5.1, we will explain why the initial number of infected profiles affects the speed of propagation logarithmically using the susceptible infected (SI) model.

5 Modeling Malware Propagation in Online Social Networks

OSN worms, like other computer worms, behave in a similar manner to biological viruses in terms of infectious disease propagation. Therefore, mathematical analyses on propagation behaviors of biological viruses can be adapted to studies of computer worms [31, 32].

In the area of epidemiology, infectious disease propagation can be modeled using either stochastic or deterministic models [1]. Stochastic models are suitable for a small-scale population, while deterministic models can be used for a large-scale population. Deterministic models should thus be used for modeling OSN worm propagation because of large sizes of OSNs. (As of December 2011, Facebook has approximately 800 million registered users around the world.)

One of the most popular differential equation models for biological worm propagation is the susceptible infected (SI) model. This model has used in several computer worm propagation models such as those in [11, 22, 32]. In this section, we discuss the SI model and existing models proposed for OSN worms.

5.1 The SI Model

The SI model is defined as follows:

$$\frac{dI(t)}{dt} = \frac{\eta}{\Omega} I(t)\big[N - I(t)\big] \tag{2}$$

In this model, N is total number of people in the population; $I(t)$ is the number of infected hosts at time t; η is the worm infectious activity rate; and Ω is the number of possible targets that can be reached by the worm. All hosts are assumed to be either vulnerable (susceptible) or infected according to the SI model. In the field of epidemiology, susceptible hosts are defined as those vulnerable to infection by the virus. Infectious hosts are those that have been infected and can infect others. A host is considered infected at time t if it had been infected before time t. Assuming that η is not a time variant variable and the initial condition is $I(t) = i_0$, the solution to (2) is as follows:

$$I(t) = \frac{i_0 N}{i_0 + (N - i_0)e^{-\eta \frac{N}{\Omega} t}} \tag{3}$$

The SI model has been used to model XSS worm propagation in OSNs by Faghani and Saidi [11]. Figure 16(a) shows the numerical results of the SI model from (3). Figure 16(b) shows the propagation trend of a real XSS worm, Samy, which attacked the MySpace network in 2005. These two graphs demonstrate that XSS worm propagation can be modeled using the SI model [11].

The simulation results in Sect. 4.2.4 show that the initial number of infected profiles i_0 affects the speed of propagation logarithmically. We can explain those results using the SI model, as follows. Given (3), to infect k per cent of the population, or kN people, we need:

$$I(t) = \frac{i_0 N}{i_0 + (N - i_0)e^{-\eta \frac{N}{\Omega} t}} \geq kN \tag{4}$$

$$t \geq \ln\left(\frac{N - i_0}{i_0 \frac{1-k}{k}}\right) \times \frac{\Omega}{\eta N} \tag{5}$$

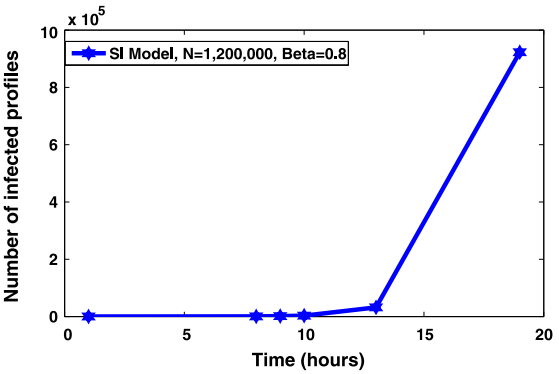
Inequality (5) suggests that the time required to infect k per cent of the population is logarithmically proportional to the initial number of infected people i_0. The simulation results presented in Sect. 4.2.4 are consistent with this suggestion: increasing the initial number of infections will decrease the time required to infect 90 % of the OSN population logarithmically.

5.2 Modeling XSS Worms

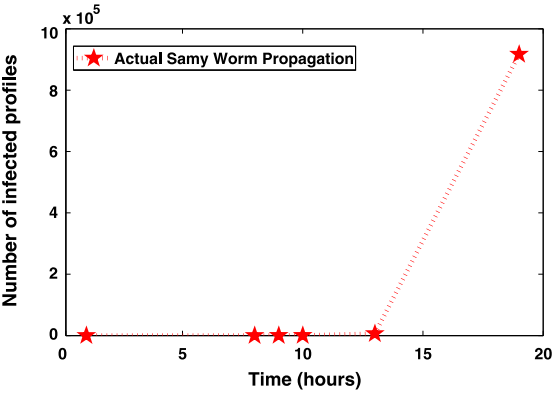
Faghani and Saidi propose the following model for characterizing XS worm propagation in OSNs [11]:

$$\frac{dI(t)}{dt} = \beta(q)\frac{I(t)}{N}\big(N - I(t)\big)^{K(q)} \tag{6}$$

Fig. 16 Samy worm



(a) SI model, numerical result



(b) Actual propagation of Samy worm

In (6), $\beta(q)$ is the rate of visiting friends in an OSN, which is a function of the visiting-friends probability q. $K(q)$ represents the sensitivity of the susceptible population to q. In general, $K(q)$ is proportional to q [11]. As q increases, members will visit their friends more often than strangers. If the number of infected friends is small, a large value of q will delay the propagation of worm in their community. Therefore, uninfected users are also sensitive to (affected by) the value of q, which $K(q)$ takes into account.

The model represented by (6) is based on the following two facts.

- First, when the visiting-friends probability q increases, the infection is more contained among friends. Less strangers would be affected by infected users. Therefore, the delay for the infection to reach other parts of the network will be longer. In other words, the rate of infection is inversely proportional to probability q.
- Second, after most of the users are being infected, the infection rate will slow down since the total number of susceptible hosts has also been decreased.

5.3 Modeling Trojan and Clickjacking Worms

Let $P(k)$ be the probability that a node in the network graph has degree k. The average degree of the network is thus $E[k] = \sum_k kP(k)$. Suppose that the fraction of infected users having degree k is $i_k(t)$. Let λ be the infection rate, which is the probability of getting infected by an infectious neighbor in a time unit. The differential equation model characterizing the infection rate of a node with degree k is as follows [18]:

$$\frac{di_k(t)}{dt} = \lambda k\big[1 - i_k(t)\big]\Theta(t) \tag{7}$$

$$\Theta(t) = \frac{\sum_n nP(n)i_n(t)}{\sum_n nP(n)} = \frac{\sum_n nP(n)i_n(t)}{E[k]} \tag{8}$$

The factor $\Theta(t)$ is the probability that a user (node) is connected by an edge to an infected user (friend) in the OSN graph. To compute $\Theta(t)$, we rely on the fact that the probability of a user having a friend with degree k is $kP(k)$ [18].

The above model is later improved in another model proposed by Boguna et al. [4], which considers the fact that the originator of an infection will not be infected again (by its children in the spanning tree of the network graph). The revised model is as follows:

$$\frac{di_k(t)}{dt} = \lambda k\big[1 - i_k(t)\big]\Theta(t) \tag{9}$$

$$\Theta(t) = \frac{\sum_n (n-1)P(n)i_n(t)}{\sum_n nP(n)} = \frac{\sum_n nP(n)i_n(t)}{E[k]} \tag{10}$$

The total number of infected nodes $I(t)$ would be:

$$I(t) = \sum_k i_k(t)P(k)N \tag{11}$$

In a recent research by Faghani et al. [8], the authors suggest an adjustment to (9) that takes into account the effects of the clustering coefficient and user behaviors as follows:

$$\frac{di_k(t)}{dt} = \lambda k\big[1 - i_k(t)\big]\Theta(t)f(c)g(p) \tag{12}$$

Functions $f(c)$ and $g(p)$ reflect the effects of the clustering coefficient and user behaviors, respectively, where c is the clustering coefficient and p is the probability that a user will click on a malicious web link. The authors leave the solutions to $f(c)$ and $g(p)$ to future work.

6 OSN Malware Countermeasures

Several malware detection and containment mechanisms for OSNs have been proposed recently. Nguyen et al. [19] propose a centralized patch distribution algorithm

which monitors the number of infected users. When the fraction of infected users reaches a pre-determined threshold, the detection system raises the alarm and sends out "treatment" patches to influential users. Influential users of a community are those having large numbers of relationships (connections) with other communities. They are thus the best candidates to distribute the "treatment" patches efficiently throughout the whole network. After receiving "treatment" patches, a user will apply them to eliminate the worm and forward them to his/her friends.

Xu et al. [28] suggest a scheme in which by monitoring a small fraction of users, the entire network can be under surveillance. An early detection will allow for effective worm containment and elimination measures.

Stein et al. describe the Facebook immune system in [23]. Their immune system performs real-time checks on every incoming and outgoing query to network. To defend against malware, their classifier identifies infected users when they send many messages flagged by the classifier or other users.

Yan et al. [29] suggest three different approaches for malware detection in OSNs. In the first approach, nodes with the highest degrees are selected and monitored for unusual messages or activities. In the second approach, the most active nodes (in terms of number of messages read and posted) in the network are monitored. In the third approach, the OSN is divided into small islands and every message exchanged between these islands is inspected.

The models and simulation results presented in Sects. 4 and 5 suggest that one of the most resource-efficient ways to defend against malware in OSNs is to detect it early within communities. The reason is that a malware will circulate among members of a community for a while before it gets a chance to move to another community, due to the high clustering property of OSNs. This approach of malware detection and containment is still an open issue for future research.

Another optimized defending mechanism is to take into account portions of the network graph which are built based on active relationships among OSN users, since active users are more likely to visit friends and execute malware code. Wilson et al. [27] show that the graph of interactions among active users is different from the graph of relationships (friendships) in an OSN.

7 Related Work

There exists research in the field of epidemiology that models the behavior of contagious diseases [4, 12, 17, 18, 21, 24]. These models can be and have been used to study the propagation of malware in online social networks [11].

Malware propagation in specific types of computer networks such as e-mail, instant messaging and mobile networks has also been well studied [5, 9, 16, 32].

Among the first works studying malware propagation in OSNs are those by Faghani and Saidi [10, 11], Yan et al. [29], and Xu et al. [28]. Faghani and Saidi [11] model the propagation of XSS worms using the SI model, and investigate malware propagation in OSNs using synthesized OSNs and based on user activities.

Yan et al. [29] use realistic network graphs in addition to realistic user activities to confirm that user activities play an important role in malware propagation in OSNs. Xu et al. [28] propose a correlation-based scheme to slow down worm propagation in OSNs. Their scheme was designed and evaluated using a real OSN graph, the Flickr network.

There exists also research on human activities in OSNs. Benevenuto et al. [3] analyze user activities on four popular OSNs (Orkut, Hi5, MySpace and LinkedIn) and provided useful information on how users behave in OSNs.

8 Chapter Summary

We discuss the characteristics of malware propagation in online social networks using analytical models and simulation results. In general, the propagation of XSS worms depends largely on users' behaviors: If OSN users visit mostly their friends rather than strangers, the worms will propagate more slowly. The highly clustered feature of social networks also helps to slow down the propagation. Increasing the initial number of infected profiles in the early stages of XSS worm propagation leads to an impressively faster propagation. Trojan worms propagate faster than XSS worms in social networks because of their inherent aggressive propagation method. Increasing the initial number of infected profiles in the early stages of Trojan worm propagation does not have considerable effects on the propagation speed. We also identify open issues for future research. First, current analytical models do not consider the network graph structure in the propagation of malware in OSNs. Second, we should exploit the high clustering structure of OSNs to detect the propagation of XSS worms early within a community, e.g., using a honeypot detection mechanism.

Acknowledgements We would like to thank Hossein Saidi, Ashraf Matrawy, Chung-Horng Lung and our anonymous reviewers for their helpful comments and discussions.

References

1. Andersson H, Britton T (2000) Stochastic epidemic models and their statistical analysis. Springer, New York
2. Barabási A, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
3. Benevenuto F, Rodrigues T, Cha M, Almeida V (2009) Characterizing user behavior in online social networks. In: Proceedings of the 9th ACM SIGCOMM conference on internet measurement conference. ACM, New York, pp 49–62
4. Boguna M, Pastor-Satorras R, Vespignani A (2003) Epidemic spreading in complex networks with degree correlations. In: Lecture notes in physics: statistical mechanics of complex networks, vol 625, pp 127–147
5. Cheng S-M, Ao WC, Chen P-Y, Chen K-C (2011) On modeling malware propagation in generalized social networks. IEEE Commun Lett 15:25–27
6. Davidsen J, Ebel H, Bornholdt S (2002) Emergence of a small world from local interactions: modeling acquaintance networks. Phys Rev Lett 88(12): 128701

7. Dekker AH (2008) Realistic social networks for simulation using network rewiring. In: Proceeding of international congress on modeling and simulation, pp 677–683
8. Faghani MR, Matrawy A, Lung C (2012) A study of malware propagation in online social networks. In: Proceeding of 5th IEEE IFIP international conference on new technologies, mobility and security. IEEE Press, New York
9. Faghani MR, Nguyen UT (2012) SoCellBot: a new botnet design to infect smartphones via social networks. In: Proceeding of 25th IEEE Canadian conference on electrical and computer engineering. IEEE Press, New York
10. Faghani MR, Saidi H (2009) Malware propagation in online social networks. In: Proceeding of the 4th IEEE international malicious and unwanted programs. IEEE Press, New York, pp 8–14
11. Faghani MR, Saidi H (2009) Social networks' XSS worms. In: Proceeding of the 12th IEEE international conference on computational science and engineering. IEEE Press, New York, pp 1137–1141
12. Griffin C, Brooks R (2006) A note on the spread of worms in scale-free networks. IEEE Trans Syst Man Cybern, Part B, Cybern 36(1):198–202
13. Grossman J (2006) Cross-site scripting worms and viruses: the impending threat and the best defense. http://www.whitehatsec.com/downloads/WHXSSThreats.pdf
14. Holme P, Beom J (2002) Growing scale-free networks with tunable clustering. Phys Rev E 65:026107
15. Lab K (2008) Detects new worm attacking MySpace and Facebook. http://www.kaspersky.com/news?id=207575670
16. Mannan M, Van Oorschot PC (2005) On instant messaging worms, analysis and countermeasures. In: Proceedings of the 2005 ACM workshop on rapid malcode. ACM, New York, pp 2–11
17. Moore C, Newman MEJ (2000) Epidemics and percolation in small-world networks. Phys Rev E 61:5678–5682
18. Moreno Y, Gomez J, Pacheco AF (2003) Epidemic incidence in correlated complex networks. Phys Rev E 68:521–529
19. Nguyen NP, Ying X, Thai MT (2010) A novel method for worm containment on dynamic social networks. In: The military communications conference. IEEE Press, New York, pp 475–478
20. Open W (2010) Application security project, OWASP top 10 project. http://www.owasp.org
21. Pastor-Satorras R, Vespignani A (2001) Epidemic spreading in scale-free networks. Phys Rev Lett 86:3200–3203
22. Staniford S, Paxson V, Weaver N (2002) How to own the internet in your spare time. In: Proceedings of 11th USENIX security symposium. USENIX Association, Berkeley, pp 149–167
23. Stein T, Chen E, Mangla K (2011) Facebook immune system. In: Proceeding of eurosys social network systems SNS. ACM, New York, pp 8:1–8:8
24. Telo Nunes A (2006) Epidemics in small world networks. Eur Phys J B, Condens Matter Phys 50(1):205–208
25. Viger F, Latapy F (2005) Efficient and simple generation of random simple connected graphs with prescribed degree sequence. In: Proceeding of the 11th conference of computing and combinatorics. Springer, Berlin, pp 440–449
26. Watts DJ (1999) Networks, dynamics, and the small-world phenomenon. Am J Sociol 105(2):493–527
27. Wilson C, Boe B, Sala A, Puttaswamy KPN, Zhao BY (2009) User interactions in social networks and their implications. In: Proceedings of the 4th ACM European conference on computer systems. ACM, New York, pp 205–218
28. Xu W, Zhang F, Zhu S (2010) Toward worm detection in online social networks. In: Proceedings of the 25th annual computer security applications conference. ACM, New York, pp 11–20
29. Yan G, Chen G, Eidenbenz S, Li N (2011) Malware propagation in online social networks: nature, dynamics, and defense implications. In: Proceedings of the 6th ACM symposium on information, computer and communications security. ACM, New York, pp 196–206

30. Yong-Yeol A, Seungyeop Kaok H, Moon S, Jeong H (2007) Analysis of topological characteristics of huge online social networking services. In: Proceeding of the 16th international conference on world wide web. ACM, New York, pp 835–844
31. Zou C, Towsley D, Gong W (2002) Code red worm propagation modeling and analysis. In: Proceedings of the ACM conference on computer and communications security. ACM, New York, pp 138–147
32. Zou CC, Towsley D, Gong W (2007) Modeling and simulation study of the propagation and defense of internet email worm. IEEE Trans Dependable Secure Comput 4(2):105–118

Estimating the Importance of Terrorists in a Terror Network

Ahmed Elhajj, Abdallah Elsheikh, Omar Addam, Mohamad Alzohbi, Omar Zarour, Alper Aksaç, Orkun Öztürk, Tansel Özyer, Mick Ridley, and Reda Alhajj

Abstract While criminals may start their activities at individual level, the same is in general not true for terrorists who are mostly organized in well established networks. The effectiveness of a terror network could be realized by watching many factors, including the volume of activities accomplished by its members, the capabilities of its members to hide, and the ability of the network to grow and to maintain its influence even after the loss of some members, even leaders. Social network analysis, data mining and machine learning techniques could play important role in measuring the effectiveness of a network in general and in particular a terror network in support of the work presented in this chapter. We present a framework that employs clustering, frequent pattern mining and some social network analysis measures to determine the effectiveness of a network. The clustering and frequent pattern min-

A. Elhajj (✉) · M. Ridley
Department of Computing, University of Bradford, Bradford, UK
e-mail: aaksac@etu.edu.tr

M. Ridley
e-mail: M.J.Ridley@bradford.ac.uk

A. Elsheikh · O. Addam · M. Alzohbi · O. Zarour · R. Alhajj
Computer Science Department, University of Calgary, Calgary, Alberta, Canada

R. Alhajj
e-mail: alhajj@ucalgary.ca

A. Aksaç · O. Öztürk · T. Özyer
Department of Computer Engineering, TOBB University, Ankara, Turkey

A. Aksaç
e-mail: aaksac@etu.edu.tr

O. Öztürk
e-mail: oozturk@etu.edu.tr

T. Özyer
e-mail: ozyer@etu.edu.tr

R. Alhajj
Department of Computer Science, Global University, Beirut, Lebanon

T. Özyer et al. (eds.), *Mining Social Networks and Security Informatics*,
Lecture Notes in Social Networks, DOI 10.1007/978-94-007-6359-3_14,

ing techniques start with the adjacency matrix of the network. For clustering, we utilize entries in the table by considering each row as an object and each column as a feature. Thus features of a network member are his/her direct neighbors. We maintain the weight of links in case of weighted network links. For frequent pattern mining, we consider each row of the adjacency matrix as a transaction and each column as an item. Further, we map entries into a 0/1 scale such that every entry whose value is greater than zero is assigned the value one; entries keep the value zero otherwise. This way we can apply frequent pattern mining algorithms to determine the most influential members in a network as well as the effect of removing some members or even links between members of a network. We also investigate the effect of adding some links between members. The target is to study how the various members in the network change role as the network evolves. This is measured by applying some social network analysis measures on the network at each stage during the development. We report some interesting results related to two benchmark networks: the first is 9/11 and the second is Madrid bombing.

Keywords Social network analysis · Terror network · Data mining · Data analysis · Knowledge discovery · Machine learning

1 Introduction

Terror is a global problem which has been severely affecting the humanity. In fact terror is mostly viewed from two perspectives. The first perspective classifies as terror all activities which are committed with the main goal of killing people, destroying the infrastructure, the economy, etc. Some activities target specific persons and others are committed without any specific target and hence are more dangerous leading to more casualties and destruction. Persons who commit such activities are known as terrorists. They are dangerous people who are the main source of threat to the whole society locally and to the humanity at large.

Terrorists are not working as individuals in isolation. They rather work in well organized groups forming networks in a way that will allow them to maximize their opportunity to hide and escape all traps which target to capture them. They try to hide because they are the target of intelligence services and law enforcement agencies. In fact, intelligence services are investing huge resources and effort to identify, capture and destroy terror networks. The fight against terror networks has been going on for long time and the success is very limited compared to the amount of resources and effort invested. Actually the interest in fighting terrorism has increased considerably after terror activities expanded from the Middle and Far East to cover the developed western countries, e.g., East Europe and USA, especially after the 9/11 attacks in USA. Research on the analysis of terror networks has concentrated on developing scientific methodologies that could help in systematically fighting terrorism, e.g., [22, 23, 25, 27].

The partial failure in the fight against terror is mostly due to two facts. The first fact is due to the second perspective where persons who are committing the killing

and destruction activities are seen as freedom fighters by some normal people who are neither joining the activities nor the fight against terrorism. The second fact is the existence of governments and countries who support these terror networks by providing all kinds of support including financial, logistic, opening training camps, etc. Even some governments try to misuse terror networks in their conflict with other governments ruling other neighboring or far away countries. Thus terror networks will continue to exist and will be anticipated to grow as long as there are conflicts between governments, nations, ethnic groups, religious groups, etc. Unfortunately, terror networks increased their effectiveness and benefit greatly from the development in technology which makes it easier for them to communicate and hide.

Internal factors include the existence of highly influential members within the network. These charismatic persons mostly take the leadership role. They try to impress and motivate other members of the network who are committing the actual terror activities. Thus, the network generally consists of two groups of members those who plan and motivate and those who execute and spread the terror and horror. Accordingly to diminish the effectiveness of a terror network it is important to consider all sources which provide strength to the network. Monitoring the usage of technology will help in identifying the identity of various members of the network. For instance, mobile communication maximizes the availability and connectivity of parties connected by the mobile technology but at the same time logs are maintained to keep track of the ongoing communication. Logs are maintained by the server and form a valuable source which could be analyzed for effective knowledge discovery. Various incidents around the world were enlightened by analyzing mobile phone logs or by utilizing advanced technology to identify the location of a suspect by tracing his/her mobile phone. Thus, the more the technology is monitored the less it is to be used by terror networks and the more they will deviate back to more difficult traditional methods to communicate and hide.

Facilitating the change of governments who support terror groups will reduce and diminish over time the effectiveness of terror networks after they lose the sources for their resources. Educating people and raising their standard of living will contribute a lot to diminish the effectiveness of terror networks. For instance, other than the leaders who put forward the ideology, AlQaeda finds more supporters in rural areas in Asia, the Middle East and North Africa. As most of the terror these days has religious or ethnic basis, the widespread of schools of thought which base their ideology on own religious interpretation should be closely controlled. Ideological leaders try to play on the emotions of the normal people and this could be avoided by the right education. Identifying these ideological leaders and eliminating them will highly contribute to decrease and may be diminish the effectiveness of a terror network.

There are various approaches to tackle the problem of terror network analysis to understand the importance of its individuals and the effectiveness of the groups within the network. Many researchers use graph and network analysis methods as measures to analyze how members of a terrorist network interact with each other and how they split into groups to plan for a terror activity, e.g., [14, 16, 22, 25, 27]. Farley [5] argues that a terror network should be viewed as a hierarchy in order

to understand the role of the various individuals within the network and the flow of information. Carley et al. [2, 3] discussed that terror networks can be dynamic and they are always changing; thus it is important to study their evolution over time and this leads to the need to investigate how the importance of individuals and groups within the network changes over time. Sparrow [23] suggested that instead of looking at the presence or absence of ties, it may be more informative to look at their strength based on task and timing. This is true because the role of individuals within the network changes dynamically and their effectiveness is affected accordingly.

The work described in this chapter tries to identify effective groups and individuals within a network in general and we concentrate on terror networks for the testing conducted. We employ social network analysis (SNA), data mining and machine learning techniques to identify the target groups and individuals. Network structure is a natural phenomenon which well represents a set of entities and their interactions. In other words, a network can be seen as a perfect match for representing various systems in diverse fields wherever it is possible to realize entities which could be connected by certain type of relationship. A network is a data structure which consists of sets of nodes (interchangeably called vertices) linked together in pairs by edges (interchangeably called links) with nontrivial topological structures [24]. A network may be visualized as a graph and may be represented for processing using a matrix or list structure. The adjacency matrix is the most commonly used representation. Various techniques in graph theory and linear algebra are valuable for network analysis and manipulation leading to a set of measures for effective network analysis. However, a network should be treated within a context in order to be analyzed for knowledge discovery within the specified context. The context is terror networks for the work described in this chapter.

The proposed framework starts with the adjacency matrix of the network and employs various SNA measures, clustering and frequent pattern mining. For clustering, every row is considered as an object and columns are the features of the objects. Each object has as values for its features the weights of the links connecting the object to its direct neighbors. In case of an unweighted network, the features will get values as either zero or one. We apply multi-objective genetic algorithm based clustering [18] to find alternative clustering solutions along the Pareto-Optimal front. Then we analyze how various individuals and groups are co-located in the same cluster across the various alternative solutions. This will lead to identifying the most influential individuals and groups within the network. Removing such individuals or groups will be investigated to study their effectiveness on the rest of the network.

The adjacency matrix will be also used as the input to Apriori [1] which is the first algorithm developed for association rules mining. By utilizing Apriori, we want to find all frequent patterns of individuals. For this purpose, each row is considered as a transaction and each column is considered as an item. Thus individuals play double role as transactions and items and this allows for smooth mining to find groups of individuals that are connected to more common other individuals. In case of a weighted network, entries in the adjacency matrix are normalized into the zero-one domain by changing into one each weight greater than zero. The groups of individuals to be returned as frequent patterns do overlap. This way, individuals

who are members of more groups will be identified as key individuals who should be eliminated to reduce or diminish the effectiveness of the network. Eliminating the latter individuals will be tested to identify potential replacements in the network; these are the next potential popular individuals.

We will apply SNA measures on the network before and after eliminating the key individuals and groups identified by clustering and/or frequent pattern mining. Actually, influential individuals and groups identified by both clustering and frequent pattern mining will receive special consideration. We will also study the effect of introducing new links on the evolution of the network to identify who could change role in the future in case new connection links are added. We will test the proposed framework by using two benchmark networks, namely the 9/11 network and Madrid bombing network.

The remaining part of this chapter is organized as follows. Section 2 describes the method that employs multi-objective genetic algorithm based clustering in order to identify effective individuals and groups within the network. Section 3 presents the frequent pattern mining based method for identifying effective individuals and groups in a network. Section 4 includes the SNA measures utilized in this study. Section 5 investigates the usefulness of eliminating individuals and groups which were identified effective. Section 6 is discussion, conclusions and future work.

2 Identifying Effective Individuals and Groups by Clustering

Clustering is the process of splitting a set of objects into groups (which may be disjoint or may overlap) such that objects in every group share high degree of similarity and objects across the groups are highly dissimilar. Thus the process starts by specifying the clustering algorithm to employ and the similarity measure to use. Various clustering algorithms are described in the literature. Each algorithm has its own advantages and disadvantages. However, most algorithms require the user to specify either the number of clusters or some parameters that will lead to the number of clusters. These need some expert knowledge and good understanding of the objects to be clustered. Further, this somehow contradicts with the basic definition of clustering as unsupervised learning technique. Thus, it is more attractive and natural to employ a technique that will need less input parameters and will decide on the number of clusters based on the input data. In other words, inspiring the number of clusters and their characteristics from the data should be the target of a comprehensive clustering algorithm.

Various similarity measures are described in the literature including both Euclidean and non-Euclidean distance measures. The choice of the similarity measure to employ in the clustering depends on characteristics of the data to be clustered. For the particular application handled in this study we decided to use the Hamming distance because all objects have characteristics with binary values and the Hamming distance is easier to compute that other sophisticated distance measures. In case of Weighted networks, Euclidean distance could be a better choice.

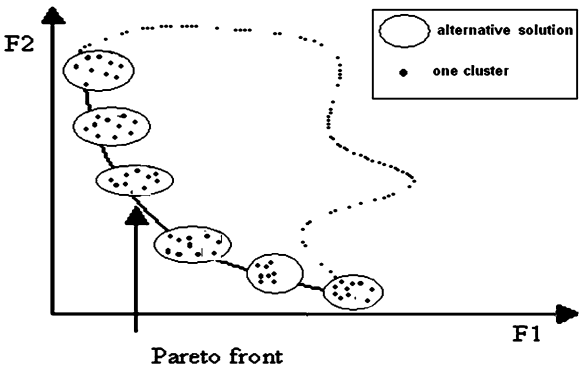


Fig. 1 Alternative solutions along the Pareto-Optimal front

For the clustering technique required for the work described in this chapter, we decided to benefit from the achievements of our research group as reported in the literature; see [12, 18, 19] for more details. The multi-objective genetic algorithm based clustering approach described in [12, 18, 19] satisfies the target by producing alternative appropriate clustering solutions. These solutions will be utilized to estimate the effectiveness of the network under investigation. In the rest of this section we first give an overview of the multi-objective genetic algorithm based clustering approach and then we will describe how the result is to be utilized to estimate the effectiveness of the network.

The multi-objective genetic algorithm based clustering approach benefits from the power of genetic algorithms to tackle large search spaces seeking appropriate solutions that fit certain predefined criteria. The basic steps of the genetic algorithm process are applied in sequence until the convergence criteria is satisfied.

The genetic algorithm based process involves a number of steps including deciding on the encoding scheme for the chromosomes (interchangeably called individuals), initialization of the chromosomes, applying cross-over and mutation to produce new chromosomes that would lead to better convergence towards the final solution, and selection based on fitness which is achieved as a combination of three objectives. The three objectives employed in the process are maximizing homogeneity within clusters, maximizing heterogeneity between clusters and minimizing the number of clusters. These objectives do conflict because for instance having every object in a separate cluster will lead to the best within cluster homogeneity but will conflict with the target of producing the minimum number of clusters. Thus, at the end of the iterative application of the steps constituting the genetic algorithm process we will get a set of alternative solutions such that none of the solutions is dominated by any of the other solutions in the set. These solutions form what is called Pareto-Optimal front as shown in Fig. 1. The number of clusters in each solution is different and objects will be differently distributed into clusters in each solution.

The encoding scheme used in this work is integer based. Every chromosome is of length N, where N is the number of objects to be clustered. And every allele (alternatively called gene) in the chromosome is assigned an integer value that represents the cluster to which the corresponding object is to be assigned. Based

on this encoding scheme, we randomly distribute objects to clusters and produce a number of initial chromosomes to be used for starting the genetic algorithm iterative process. For the work described in this chapter, we decided arbitrarily on using 50 initial chromosomes. In each chromosomes objects are randomly distributed into clusters and the homogeneity of each cluster could be determined by considering its objects to compute the total within cluster variation (TWCV), which calculates the intra-cluster distance of the cluster by the following formula:

$$TWCV = \sum_{n=1}^{N}\sum_{d=1}^{D} X_{nd}^2 - \sum_{k=1}^{K}\frac{1}{Z_k}\sum_{d=1}^{D} SF_{kd}^2 \tag{1}$$

where $X_1, X_2, \ldots, X_N$ are N objects, X_{nd} denotes feature d of pattern X_n ($n =$ 1 to N); K is the number of clusters; SF_{kd} is the sum of the dth features of all the patterns in cluster $k(G_k)$; Z_k denotes the number of patterns in cluster $k(G_k)$. Actually, SF_{kd} is computed as:

$$SF_{kd} = \sum\nolimits_{\overrightarrow{x_n} \in G_k} X_{nd} \quad (d = 1, 2, \ldots, D) \tag{2}$$

For separateness, we used the average to centroid linkage based inter-cluster separability formula described next, where P and R denote clusters and $|P|$ and $|R|$ are the cardinalities of the aforementioned clusters; $d(x, y)$ is the similarity (distance) metric where $x \in P$, $y \in R$, and $P \neq R$.

Average to centroid linkage is the distance between the members of one cluster to the other cluster's representative member, i.e., the centroid. It is computed as:

$$D(P, R) = \frac{1}{|P| + |R|}\left[\sum_{x \in P} d(x, v_R) + \sum_{y \in R} d(y, v_P)\right] \tag{3}$$

Based on the computed linkage value D, the total inter-cluster distance (TICD) is computed as:

$$TICD = \sum_{k=1}^{K}\sum_{l=k+1}^{K} D(k, l) \tag{4}$$

Cross-over is an operation that leads to the evolution of the population towards the final stable state. It is supported by mutation to help in quick convergence. There are several cross-over operators in use. They mostly take two of the existing chromosomes and produce one or more new chromosomes that may have better fitness than their parents. We apply a variation of arithmetic cross-over in order to produce the number of clusters as a by-product of the process without requiring it as an input parameter. This leads to more natural clustering with less input parameters. The arithmetic crossover method used in the testing works as follows [10, 12, 17].

Consider two chromosomes each of length N:

$$C_1 = \left(c_1^1, \ldots, c_N^1\right) \quad \text{and} \quad C_2 = \left(c_1^2, \ldots c_N^2\right)$$

where N is the number of objects to be clustered.

Applying the crossover operator on C_1 and C_2 generates two offspring:

$$H_1 = \left(h_1^1, \ldots, h_i^1, \ldots, h_N^1\right) \quad \text{and} \quad H_2 = \left(h_1^2, \ldots, h_i^2, \ldots, h_N^2\right)$$

where for i=1 to N, $h_i^1 = (Int(\lambda c_i^1 + (1-\lambda)c_i^2) \bmod K) + 1$ and $h_i^2 = (Int(\lambda c_i^2 + (1-\lambda)c_i^1) \bmod K) + 1$; here $Int()$ is a function that produces the integer value of the argument; the modulus function is applied to guarantee that the produced cluster number is always mapped into the range $[1, K]$. In the process, λ varies with respect to the produced number of generations, as non-uniform arithmetical crossover. Further, clusters' numbers are readjusted in case any of the values in the range $[1, K]$ is skipped. For instance, if the new chromosome includes clusters numbered 1, 2, 3, 5, and 7, where 4 and 6 are skipped then the clusters are renumbers such that 5 becomes 4 and 7 become 5 to reflect the fact that there are only 5 clusters in the chromosome.

After completing the cross-over operation which started with 50 chromosomes and considered them as 25 pairs, the number of chromosomes increases to 75. Then the fitness of each chromosome is measured as the sum of three values, namely the average homogeneity of its clusters and the average separateness between the clusters (both computed using the formulas given earlier in this section) as well as the average size of the clusters. Finally, the 75 chromosomes are ranked based on their fitness in descending order and the best 50 chromosomes are kept for the next iteration. The process continues until one of two stopping conditions is satisfied, either we reach 500 iterations or the improvement between consecutive iterations is very small compared to the average improvement during the previous iterations.

The final set of chromosomes are ranked based on their fitness and the best five chromosomes are considered as the final clustering solution to be used in the analysis that targets estimating the importance of the various individuals and groups with the input network. However, to validate the latter selection process, we apply cluster validity indexes on the top 10 individuals from the final solution produced by the genetic algorithm process. The outcome from the validity analysis will confirm the appropriateness of the selected solutions, which are the five solutions favored by the majority of the validity indexes [18].

The best five solutions will be analyzed further by applying the following process. Each clustering solution is analyzed separately first and then all the solutions are analyzed collectively. Within each solution, terrorists are individually checked to find how each terrorist is close to the centroid of his/her cluster. We build a hierarchy from each cluster where the level of each terrorist is determined based on his/her distance from the centroid. Terrorists at distance i from the centroid are placed at level i in the hierarchy. The child parent relationship is determined as follows: each terrorist at level $i \geq 1$ is connected to the closest terrorist at level $i-1$. This way terrorists closer to the centroid are placed towards the root and terrorists at the boundary of the cluster form the leaves of the hierarchy. The relevance of terrorists increases as they are closer to the centroid and hence to the root of the constructed hierarchy.

We build the hierarchies for all the clusters in the five alternative solutions selected above. Then the hierarchies are processed to determine a rank for each terrorist based on his/her level in each of the constructed hierarchies. For this purpose

we use a simple ranking function which is inversely proportional to the level of the terrorist in each hierarchy in which he/she exists.

$$Rank(r) = \frac{\sum_1^m (\frac{1}{i})}{p} \tag{5}$$

where m is the number of hierarchies in which terrorist r exists and p is the total number of hierarchies. We divide by the total number of hierarchies to avoid giving higher rank to terrorists who exist in less number of hierarchies.

By considering all the produced hierarchies we are also able to find effective groups of terrorists. For this purpose, we identify groups of terrorists forming a subhierarchy which repeats in a majority of the hierarchies. We rank subhierarchies based on the number of hierarchies in which they repeat and then we select the subhierarchies that repeat in more than the average as the ones to be considered as constituting more effective groups of terrorists.

3 Study Effectiveness by Frequent Pattern Mining

Given a set of transactions each contains a set of items, frequent pattern mining is a technique that determines non-empty sets of items such that each considered set is subset from at least a prespecified number of transactions. The importance and effectiveness of a given set of items is directly proportional to the number of transactions that include the set. An important set of items in a supermarket setting is the set of items purchases by most of the customers because each transaction reflects the purchase pattern of one customer. For a terror network, on the other hand, a set of members is important if they are directly connected to the same large set of members in the network. The larger the latter set is the more effective will be the former set. Therefore, we want to employ a systematic way that will lead to all frequent sets of members from a given terror network. The approach described here is not specific for terror network, it is general enough such that it could be applied to any network.

Several frequent pattern mining algorithms are described in the literature, e.g. [1, 9]. They take the same input and produce the same result. But, they only differ in the way they process the data and hence in their performance. The input to any frequent pattern mining algorithm is a set of transactions such that each transaction consists of a set of items. The transactions and items terminology was inspired from the first application of the methodology which is market basket analysis. However, the methodology is general enough to successfully serve any application domain which contains two sets of entities that are correlated by a many to many kind of relationship. Accordingly, the frequent pattern mining methodology has been successfully applied to a wide range of domains from document analysis (documents are transactions and words are items) to the study of drug-protein interactions (drugs are transactions and the proteins they handle are items), etc.

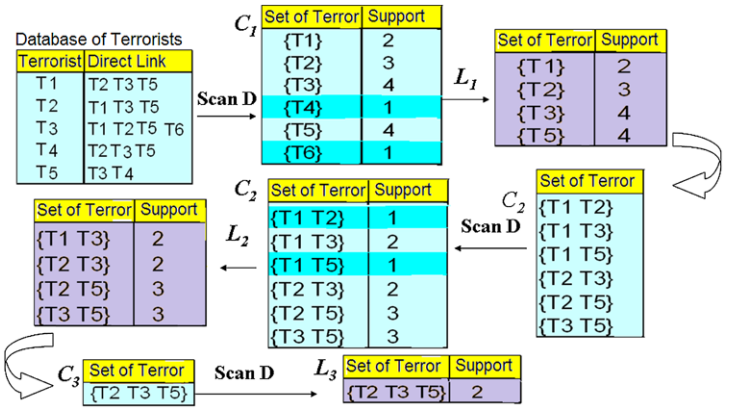


Fig. 2 Illustrating the frequent pattern mining steps of Apriori

The basic step to be accomplished before applying the frequent pattern mining methodology to a new application domain is to decide on the items and the transactions. Then the process becomes straightforward by applying any of the available frequent pattern mining algorithms. For the application domain studied in the work described in this chapter, we map the network into transactions and items by considering the adjacency matrix. Every row corresponds to one transaction and every column is one item. That is, terrorists play the role of transactions and items. One terrorist is considered to correspond to a transaction and all terrorists directly linked to him/her are said to be the items constituting the transaction. From the model, we will be interested in frequent sets of terrorists.

A set of terrorists is said to be frequent if it includes terrorists who are linked directly and concurrently to the same other terrorists such that the number of the latter terrorists is larger than a predefined minimum threshold. The threshold is mostly specified by an expert or may be determined from the data by applying machine learning techniques like hill-climbing. To determine frequent sets of terrorists we apply the Apriori algorithm. The main reason for using Apriori algorithm is its simplicity and because the networks we are going to process are relatively small. Hence the performance is not an issue to worry about.

The basic steps of the Apriori algorithm are illustrated in Fig. 2 on a small example database of terrorists. The algorithm starts with the database where there are six terrorists but only five of them have links to other terrorists leading to five transactions. The first step determines the frequency of each terrorist in the given database by counting, to how many other terrorists the given terrorist is linked. At each iteration, Apriori first finds the candidate sets of terrorists which should be checked to determine whether they are frequent. Each candidate set is denoted as C_i and each frequent set is denoted as L_i for step i. A candidate set contains all possible sets of a particular size and they are derived from the frequent sets which were determined in the previous iteration. Then the support of each candidate set is determined as the number of transactions that contain all the members in the set. Only candidate

sets that satisfy a predefined minimum support threshold are kept as frequent sets; all other sets are eliminated. The process continues until either one frequent set is obtained as in the example shown in Fig. 2 or no more frequent sets could be found. One database scan is needed each time it is required to find the frequencies of newly generated candidate frequent sets.

Frequent sets of terrorists are determined with minimum support set to a value to be determined based on the sparseness of the network. For sparse networks, we set the support threshold to lower value closer to one to be able to analyze all cases starting with terrorists linked to only one other terrorist. However, for dense networks we target more connected groups and hence we set the support threshold to a higher value. Terrorists linked to less number of other terrorists may also reveal important information regarding the whole terror network and may be regarding the strategy implemented by the terror organization behind the network.

Frequent sets of terrorists are ranked by their frequency in descending order. It is anticipated that smaller sets will rank higher in general because all subsets of a frequent set are also frequent and the frequency is a non-increasing function as the size of the set increases. In other words, all subsets of a given set s will have at least same frequency as s.

The ranked sets are utilized to estimate the importance of individual terrorists as well as the effectiveness of groups of terrorists as follows. The importance of an individual terrorist is directly proportional to the number of frequent sets in which he/she exists. Further, an individual is expected to be more effective on others with whom he/she coexist in more sets with higher frequency. In other words, we first find the most influential terrorists. Next, we determine who they could influence. Here, the same methodology is applied as group effectiveness is concerned. Thus, subsets of terrorists who coexist together in more frequent sets are determined. Such a group should be considered more effective than others. The importance of an individual or a group could be quantified as the average frequency of all the sets in which the terrorist or the group exists. Further, a terrorist or a group is considered more influential if both the clustering and the frequent pattern mining techniques agree on the level of effectiveness. Eliminating such terrorists or groups from the network may lead to quicker collapse of the whole network.

4 Social Network Analysis Measures

The structure of a network is a set of nodes and a set of links connecting part or most of the nodes. This property allows for effective analysis of the two sets separately or jointly, in order to discover some valuable information. Benefitting from the fundamentals of graph theory, statistics and linear algebra researchers have defined a set of measure that could be employed to determine the importance of individual nodes/links as well as the effectiveness of a group of nodes/links. Most of the measures concentrate on the connectivity of the network and its completeness. For instance, nodes with high degree are more effective. In addition they contribute more

to the completeness of a network. Other nodes or links may bridge two or more parts of a network. Hence, their removal may lead to network partitioning. We apply SNA measures to the original terror network in order to identify effective individuals and groups [4, 8, 15, 21]. The outcome will help in validating the discoveries reported by the two techniques described in Sects. 2 and 5.

One of the simple node based SNA measures is degree centrality which is roughly defined as the number of links directly connected to a node [6, 7, 11, 26]. The importance of a node is directly proportional to its degree centrality. A leader is expected to be connected to almost every other node. As links in a network may be either directed or undirected, it is possible to differentiate between two ways for computing degree centrality. For an undirected network the degree centrality is the total number of links connected to the node. However, for a directed network, it is possible to have two types of links connected to each node leading to in-degree and out-degree. In-degree of a node a is the number of links that start at node a and end at another node in the network. Out-degree of node a is the number of links that start at any other node and end at node a. As a result the degree of node a will be the sum of its in-degree and out-degree. For an undirected network the maximum degree is $N - 1$, where N is the number of nodes. On the other hand, a directed network has the upper bound for both in-degree and out-degree as $N - 1$. A network member with a high degree centrality value could be the leader or "hub" in a network.

Degree based analysis may lead to identify network members who play the role of authority or hub. The former represent members who have high in-degree and the latter refer to members who have high out-degree connecting them to authorities [8, 26]. In other words, an authority is a receiver and a hub is a distributor to authorities. A network member is said to be authority central if its in-degree is from other members with high out-degree [13]. Kleinberg [13] proposed an iterative algorithm for measuring authority and hub degrees of each member in a network.

The importance of members of a network may be also determined by measuring the eigenvector centrality which is another node based SNA measure. A member is considered more effective when it is connected to other effective members in the network [7, 26]. Eigenvector centrality is computed based on the adjacency matrix of the network under investigation. The centrality score of each member is computed based on the sum of the scores of all other members as shown in Eq. 6.

$$X_i = \frac{1}{\lambda} \sum_{j \in M(i)} x_j = \frac{1}{\lambda} \sum_{j=1}^{n} A_{i,j} x_j \tag{6}$$

where A is the adjacency matrix of the network, x_i is the score of member i, $M(i)$ is set of members connected to member i, N is the total number of members in the network, and λ is a constant value known as the eigenvalue. A given adjacency matrix may lead to a number of eigenvalues. Each eigenvalue may have a corresponding eigenvector. For eigenvector centrality, we are interested in the eigenvector that corresponds to the largest positive eigenvalue. Actually, eigenvector centrality is an important measure for showing the importance of individual members in a network. A member with high eigenvector centrality can spread information much faster com-

pared to other members as he/she is well-connected to other well-connected members in the network.

Clustering coefficient is another important measure that reveals network structure related information by considering the links between adjacent members in the network. For a given member in a network, clustering coefficient measures how close his/her neighbors are to being a clique (complete subgraph). Direct neighbors based clustering coefficient (CC) is computed as follows.

$$CC(a) = \frac{2|E|}{d(a) \times (d(a) - 1)} \tag{7}$$

where $|E|$ is the number of links connecting members who are direct neighbors of a, and $d(a)$ is the number of links directly connected to a.

While degree, eigenvector, clustering coefficient, authority and hub are node based measures, other SNA measures depend on the links. For instance, closeness and betweenness are two measures that depend on the shortest path between nodes. A shortest path is a sequence of links that satisfy the following criteria: (1) it starts at a node and ends at another node, (2) no link is considered more than once in the path, (3) no node is considered more than once in the path, and (4) it is the sequence that contains the least number of links. Every path has a length which is the total number of links constituting the path in case of unweighted network; and it is the sum of weights of the links forming the path in case of a weighted network.

The shortest path concept (which is sometimes called the geodesic distance) can be used to define closeness centrality of node a based on the sum of the lengths of the shortest paths that connect a to every other node in the network. The latter sum is divided by the number of nodes minus one to get the normalized closeness centrality measure. The importance of members in a network is inversely proportional to the closeness centrality. The most effective members in a network have their normalized closeness centrality close to one.

All the shortest paths in a network can serve another SNA measure, which is the betweenness centrality. The relevance and importance of any member in a network may be measured by considering the number of shortest paths, in which the member appears. Effective members appear in more shortest paths compared to other members in the network. It is also possible to consider links in connection with shortest paths. A link is said to be more important when it belongs to more shortest paths in the network. A member with high betweenness centrality value may act as a gatekeeper or "broker" for communication or information passing in the network. Another variation of betweenness is group betweenness centrality [20]. Each group member is tested for shortest path in that case.

5 Testing the Effectiveness of Terror Networks

To demonstrate the effectiveness and applicability of the framework described in this chapter, we conducted experiments using data related to two terror networks, namely 9/11 and Madrid bombing. The 9/11 data set consists of 63 members of the

terror network who were involved in 9/11 attacks. There are 153 links in the 9/11 network; these links connect members in the network and they were constructed based on various relationships, including whether the suspects have communicated, they are relatives, they share the place of origin, or they are roommates. The Madrid Bombing data set consists of 67 nodes and 89 links. The links were constructed by applying the same measures enumerated above for the 9/11 data set.

We derived the adjacency matrix of the 9/11 terror network. Each of the 63 members gets as its features the links to his direct neighbors in the network. We utilized the 63 members with their features as input to the multi-objective genetic algorithm based clustering technique. We utilized the Manhattan distance measure to compute the similarity between objects. It is interesting to discover that Mohamed Atta is the closest to the centroid of every cluster he belongs to in all the five top solutions returned by the clustering technique. In addition, Saeed Alghamdi, Essid Sami Ben Khemais, Hani Hunjor, Djamal Beghal, Nawaf Al hazmi and Marvan Al-Shehhi are strong examples of terrorists who were reported as effective members of the 9/11 network. As another evidence of their effectiveness these terrorists were closer to the root of the hierarchies derived from their clusters.

We further run the Apriori algorithm using the adjacency matrix of the 9/11 terror network. We discovered all frequent sets of terrorists and investigated their effectiveness. Because the network is not dense we kept minimum support value arbitrarily at 3, which means a set of terrorists is considered frequent if its members are concurrently related (direct neighbors of) to at least three terrorists. The reported results for the 9/11 terror network revealed high overlap with the results reported by the clustering approach. This confirms the importance of the members listed above.

We applied the clustering technique and the frequent pattern mining approach on the Madrid bombing data set. The most effective terrorists reported by the framework are Jamal Zougam, Jamal Ahmidan, Serhane ben Abdelmajid Fakhet, Redouan Al-Issar, Mohamed Bekkali and Abdennabi Kounjaa.

The above results for both networks are well supported by the SNA measures described in Sect. 3. For instance, in the 9/11 network Mohamed Atta, Hani Hunjor, Essid Sami Ben Khemais, Marvan Al-Shehhi and Nawaf Al-Hazmi are the top five based on degree centrality; Mohamed Atta, Essid Sami Ben Khemais, Djamal Beghal, Zaoarias Moussaoui and Hani Hunjor ranked the top five with high betweenness values; Raed Hijazi, Hamza Alghamdi, Saeed Alghamdi, Nabil Al-Marabh and Amed Alnami have the best closeness values.

To further realize the effectiveness of the most influential terrorists in the network, we removed Mohamed Atta from the network by eliminating the row and column that represent his node in the adjacency matrix. We repeated the above analysis and determined that the effectiveness of Ziad Jarrah increased. The elimination of Mohamed Atta also affected the structure of the network as another evidence of his importance as a key member of the 9/11 network. To further investigate effective members in the network we removed both Marvan Al-Shehhi and Ziad Jarrah. We realized that the network got more distorted and the importance of Nawaf Al-Hazmi increased. More detailed investigation will be conducted in the future to closely analyze the importance of each member and each group. Groups will be utilized directly from the sub hierarchies in the clusters and from the frequent sets of terrorists.

6 Discussion, Conclusions and Future Work

Estimating the effectiveness of a network is an essential task that once accomplished successfully could reveal variety of facts to be utilized in studying the network. Terror networks need special treatment when it comes to study their effectiveness because there are various factors that could play important role in shaping the effectiveness of the network. Some factors are internal to the model and could be analyzed and studied by developing a scientific methodology as we did in this chapter. However, external factors are very volatile and are hard to control and eliminate. While scientific techniques like the one described in this chapter can help in analyzing a terror network, the external factors may lead to a dynamic and uncontrollable network. For instance, terror organizations use global changes in their propaganda to hire new members based on ethnicity, religion, poverty, discrimination, etc. So, it is not an easy task to produce a complete analysis of the effectiveness of a terror network especially in this era where terror networks are growing into international organizations; they are spreading like cancer. The effectiveness of terror networks are alleviating as the conflicts are spreading between various ethnic and religious groups, especially in the Middle East. Misunderstandings of religion have always created problems and conflicts. It is important to watch out for unknowledgeable scholars who play with the emotions of the youth and motivate them to turn into terrorist candidates.

The framework described in this chapter provides a systematic approach to study the importance of individuals and groups within a network. The two techniques constitute the core of the proposed framework. They clearly highlight the most influential individuals and groups by considering alternative clustering solutions and frequent patterns produced by analyzing the adjacency matrix of the given network. Building hierarchies from the clusters based on distance from the centroid helps a lot in determining the potential influence of every member in a network as well as the influence of small and large groups within the network. Further, studying the overlap between frequent sets of terrorists allowed us to determine the most effective individuals and groups. The two results complement each other and turn the whole framework into a robust system for estimating the effectiveness of a network.

The discoveries reported in this chapter are very encouraging and need to be further emphasized and supported by more extensive testing. We will utilize other terror networks which have been published in the literature, including Bali Bombing, World Trade Center, London Bombing, etc. We will employ group betweenness centrality [20]. We will also test the applicability of the proposed framework to normal networks from various applications including gene regulatory networks, gene-gene, protein-protein interaction networks, protein-disease, protein-drug and disease drug networks. We want to complete comprehensive testing by analyzing the importance of every individual in the network. We want also to benefit from the hierarchies produced from the outcome of the clustering approach to study the influence of every sub hierarchy. We will emphasize more on sub hierarchies common to most of the alternative clustering solutions. In the study described in this chapter, we used only the top five alternative clustering solutions; we want to extend the set of alternative

solutions to be considered to include more of the solutions along the Pareto-Optimal front. Considering all alternative solutions needs more computing power and more effort. We are working on acquiring a powerful machine that will facilitate the more comprehensive and detailed testing. We will also build a classifier framework to determine the importance of individuals and groups based on their characteristics to be captured by the classifier framework by considering some training data. Finally, we will work on tools and techniques that will allow us to watch the effectiveness of a terror network by benefitting from the Web technology and social media. We will utilize open source intelligence in the process.

References

1. Agarwal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: ACM SIGMOD international conference on management of data
2. Carley KM (2003) Dynamic network analysis. In: Breiger R, Carely KM (eds) The summary of the NRC workshop on social network modeling and analysis. National research council
3. Carley KM, Reminga J, Kamneva N (2003) Destabilizing terrorist networks. In: NAACSOS conference proceedings, Pittsburgh, PA
4. Carrington PJ, Scott J, Wasserman S (2005) Models and methods in social network analysis. Cambridge University Press, Cambridge
5. Farely DJ (2003) Breaking Al Qaeda cells: a mathematical analysis of counterterrorism operations. Stud Confl Terror 26:399–411
6. Freeman LC (1977) A set of measures of centrality based on betweenness. Sociometry 40:35–41
7. Freeman LC (1980) The gatekeeper, pair-dependency and structural centrality. Qual Quant 14(4):585–592
8. Freeman LC, White DR, Romney AK (1992) Research methods in social network analysis. Transaction Publishers, New Brunswick
9. Han J, Pei J, Yin Y (2000) Mining Frequent patterns without candidate generation. In: ACM SIGMOD international conference on management of data
10. Herrera F, Lozano M, Verdegay JL (1998) Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. Artif Intell Rev 12(4):265–319
11. Jialun Q, Xu JJ, Daning H, Sageman M, Chen H (2005) Analyzing terrorist networks: a case study of the global Salafi Jihad network. In: Proceedings of IEEE international conference on intelligence and security informatics, Atlanta GA, pp 287–304
12. Kaya M, Alhajj R (2008) Multi-objective genetic algorithms based automated clustering for fuzzy association rules mining. J Intell Inf Syst 31(3):243–264
13. Kleinberg JM (1998) Authoritative sources in a hyperlinked environment. In: Proceedings of the ninth annual ACM-SIAM symposium on discrete algorithms, pp 668–677
14. Klerks P (2001) The network paradigm applied to criminal organizations. Connections 24(3)
15. Knoke D, Yang S (2008) Social network analysis, series: quantitative applications in social sciences. Sage, Thousand Oaks
16. Krebs V (2002) Mapping networks of terrorist cells. Connections 24(3):43–52
17. Michalewicz Z (1992) Genetic algorithms + data structures = evolution programs. Springer, Berlin
18. Özyer T, Alhajj R (2006) Achieving natural clustering by validating results of iterative evolutionary clustering approach. In: Proc. of IEEE international conference on intelligent systems, pp 488–493
19. Peng P Nagi M et al (2011) From alternative clustering to robust clustering and its application to gene expression data. In: Proc. of IDEAL, LNCS. Springer, Norwich

20. Puzis R, Elovici Y, Dolev S (2007) Finding the most prominent group in complex networks. AI Commun 20(4):287–296. 2007
21. Scott J (1998) Trend report: social network analysis. Sociology 109–127
22. Shaikh MA, Wang J (2006) Discovering hierarchical structure in terrorist networks. In: Proceedings of the international conference on emerging technologies, pp 238–244
23. Sparrow MK (1991) The application of network analysis to criminal intelligence: an assessment of the prospects. Soc Netw 13(3):251–274
24. Strogatz SH (2002) Exploring complex networks. Nature 410:268–276
25. Tsvetovat M, Carley KM (2005) Structural knowledge and success of anti-terrorist activity: the downside of structural equivalence. J Soc Struct 6(2)
26. Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press, Cambridge
27. Xu J, Chen H (2005) CrimeNet explorer: a framework for criminal network knowledge discovery. ACM Trans Inf Syst 23(2):201–226