Data-Oriented Models of Parsing and Translation

Mary Hearne

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University

School of Computing

Supervisor: Dr. Andy Way

January 2005

Abstract

The merits of combining the positive elements of the rule-based and data-driven approaches to MT are clear: a combined model has the potential to be highly accurate, robust, cost-effective to build and adaptable. While the merits are clear, however, how best to combine these techniques into a model which retains the positive characteristics of each approach, while inheriting as few of the disadvantages as possible, remains an unsolved problem. One possible solution to this challenge is the Data-Oriented Translation (DOT) model originally proposed by Poutsma (1998, 2000, 2003), which is based on Data-Oriented Parsing (DOP) (e.g. (Bod, 1992; Bod et al., 2003)) and combines examples, linguistic information and a statistical translation model.

In this thesis, we seek to establish how the DOT model of translation relates to the other main MT methodologies currently in use. We find that this model differs from other hybrid models of MT in that it inextricably interweaves the philosophies of the rule-based, example-based and statistical approaches in an integrated framework.

Although DOT embodies many positive characteristics on a theoretical level, it also inherits the computational complexity associated with DOP. Previous experiments assessing the performance of the DOT model of translation were small in scale and the training data used was not ideally suited to the task (Poutsma, 2000, 2003). However, the algorithmic limitations of the DOT implementation used to perform these experiments prevented a more informative assessment from being carried out. In this thesis, we look to the innovative solutions developed to meet the challenges of implementing the DOP model, and investigate their application to DOT. This investigation culminates in the development of a DOT system; this system allows us to perform translation experiments which are on a larger scale and incorporate greater translational complexity than heretofore. Our evaluation indicates that the positive characteristics of the model identified on a theoretical level are also in evidence when it is subjected to empirical assessment. For example, in terms of exact match accuracy, the DOT model outperforms an SMT model trained and tested on the same data by up to 89.73%.

The DOP and DOT models for which we provide empirical evaluations assume context-free phrase-structure tree representations. However, such models can also be developed for more sophisticated linguistic formalisms. In this thesis, we also focus on the efforts which have been made to integrate the representations of Lexical-Functional Grammar (LFG) with DOP and DOT. We investigate the usefulness of the algorithms developed for DOP (and adapted here to Tree-DOT) when implementing the (more complex) LFG-DOP and LFG-DOT models. We examine how constraints are employed in these models for more accurate disambiguation and seek an alternative methodology for improved constraint specification. We also hypothesise as to how the constraints used to predict both good parses and good translations might be pruned in a motivated fashion. Finally, we explore the relationship between translational equivalence and limited generalisation reusability for both the tree-based and LFG-based DOT models, focussing on how this relationship differs depending on which formalism is assumed.

Acknowledgements

Firstly, I would like to thank my supervisor, Andy Way. It was thanks to him that I became interested in research in the first place, and he has been a constant source of encouragement, inspiration and common sense throughout the course of my studies – in short, the ideal supervisor!

My visit to the University of Amsterdam during May and June of 2003 was a highlight of my time as a PhD student. I would like to thank Khalil Sima'an and Rens Bod for a stimulating and extremely rewarding six weeks, and for their continued interest and support.

I thank all the members of the National Centre for Language Technology, and the staff and postgraduate students at the School of Computing and School of Applied Languages and Intercultural Studies at DCU for their helpful comments whenever I presented my work. In particular, I thank Josef van Genabith for his unfailing interest and enthusiasm. Special thanks go to Nano, Mick and Ruth, and especially Aoife, for always listening to my ideas and answering my never-ending stream of silly questions, and to Declan, whom I've really enjoyed working with.

I thank Tracy Holloway King and Martin Forst for providing me with data. I acknowledge the DCU School of Computing for providing financial support without which this work would not have been possible. This work was also partly funded by an Albert College Senior Research Fellowship awarded to Andy Way.

Of course, I also acknowledge everyone who has helped to keep me from cracking up over this past year. I mention Andy here – again! – for being the kind of supervisor who realises that there is more to life than just work. Thanks to Cathal, Tom and Aoife for finishing before me and proving that it is possible to survive writing up with your mental health intact! Thanks to Nano for always being on the same timescale as me so I haven't had to write up on my own, and to Michelle who has helped keep us both (relatively) calm – don't worry, we'll return the favour! Thanks to Mick and Barry for always being in good spirits when we're all in working on the weekend, and to everyone in CAPG for keeping me sane.

Special thanks to my parents, Michael and Mary, for always supporting me and instill-

ing me with the belief that I could achieve anything I set my mind to, to Kay for being the kind of older sister who always set me a high standard to aim for but is also good craic and down-to-earth, and to Liam for making sure I never take myself too seriously. Thanks also to Gillian and Teresa for not forgetting me, even though I hardly ever get to come home these days.

Finally, special thanks to Pete for always knowing when to encourage me to relax and take a break, and when to make me get my act together and get some work done – without him, I would have finished this thesis either a lot sooner or not at all!

Contents

Chapter 1

Introduction

There are two main paradigmatic approaches to the automation of the translation process. Broadly speaking, rule-based systems translate by following a set of instructions provided by linguistic experts, whereas data-driven systems learn from example sentences translated by humans. Increasingly, machine translation (MT) research is converging towards hybrid models. For example, knowledge for rule-based MT can be induced automatically from corpora, while data-driven methods are increasingly incorporating linguistic information. The merits of combining the positive elements of the rule-based and data-driven approaches to MT are clear: a combined model has the potential to be highly accurate, robust, cost-effective to build and adaptable. While the merits are clear, however, how best to combine these techniques into a model which retains the positive characteristics of each approach, while inheriting as few of the disadvantages as possible, remains an unsolved problem to which many solutions are possible.

One possible solution to the challenge of developing an optimal hybrid MT framework is the Data-Oriented Translation (DOT) model (Poutsma, 1998, 2000, 2003), which is based on Data-Oriented Parsing (DOP) (e.g. (Bod, 1992; Bod et al., 2003)) and combines examples, linguistic information and a statistical translation model. Studies of this model carried out previously (Poutsma, *op cit.*) leave some important research questions unanswered.

In this thesis, we seek to establish how the DOT model of translation relates to the other main MT methodologies currently in use. We find that this model differs from other

approaches in that it is not allied to any one of rule-based, example-based and statistical MT over the others, but rather inextricably interweaves the philosophies of all three in an integrated framework. In short, in the DOT model, none of the three elements – linguistics, statistics and examples – plays a more or less important role than the others. We find that the unique characteristics of this approach to translation render it worthy of empirical investigation.

Although DOT embodies many positive characteristics on a theoretical level, it also inherits the computational complexity associated with DOP. Previous experiments assessing the performance of the DOT model of translation were small in scale and the training data used was not ideally suited to the task (Poutsma, 2000, 2003). However, the algorithmic limitations of the DOT implementation used to perform these experiments prevented a larger-scale, more informative assessment from being carried out. In this thesis, we look to the innovative solutions developed to meet the challenges of implementing the DOP model, and investigate their application to DOT. This investigation culminates in the development of a DOT system which allows for a more intensive evaluation than heretofore.

Thanks to this new, more sophisticated DOT implementation, we are in a position to perform translation experiments which are on a larger scale and incorporate greater translational complexity than before. In this thesis, we rigorously assess the capabilities of the DOT model using up-to-date evaluation techniques. In doing so, we seek to ascertain whether the positive characteristics of the model identified on a theoretical level are also in evidence when it is subjected to empirical evaluation.

In order to address the practical questions which arise regarding the implementation and evaluation of the DOT model, we present a detailed account of the published work – both theoretical and practical – on DOP. Furthermore, we build a working DOP system using the techniques we apply when implementing DOT. Of course, as we intend this parser to form the core technology behind our translation system, we replicate previous DOP experiments (Bod and Kaplan, 2003) in order to verify that our results are consistent. In addition, we present a thorough evaluation of parsing experiments on both English and French data in the interests of further establishing the characteristics of the DOP model

itself.

The DOP and DOT models for which we provide empirical evaluations assume context-free phrase-structure tree representations. However, data-oriented models of parsing and translation can also be developed for more sophisticated linguistic formalisms. In this thesis, we also focus on the efforts which have been made to integrate the representations of Lexical-Functional Grammar (LFG) – which associate with each phrase-structure tree an attribute-value matrix encoding lexical and functional information – with the DOP and DOT models of parsing and translation.

We outline the theoretical and empirical work which has been carried out to date on the LFG-DOP model (e.g. (Bod and Kaplan, 1998, 2003)). We show how parameter re-estimation techniques developed for the DOP model which assumes phrase-structure trees (Sima'an and Buratto, 2003) can be applied to the LFG-DOP model (Hearne and Sima'an, 2003). We also study how constraints are employed for more accurate disambiguation and seek an alternative methodology for improved constraint specification. Again, we look to the innovative solutions developed to meet the challenges of implementing the DOP model, and investigate how useful they are when implementing the (more complex) LFG-DOP model.

Finally, we describe the research that has been carried out on the LFG-DOT translation model, all of which is theoretical in nature (Way, 1999, 2001). We bring together the knowledge acquired through implementation of the tree-based DOT model and study of the complexities of extending our DOP system with LFG representations to suggest how LFG-DOT might best be implemented. Furthermore, we explore the relationship between translational equivalence and limited generalisation reusability for both the tree-based and LFG-based DOT models, focussing on how this relationship differs depending on which formalism is assumed. In addition, we hypothesise as to how the constraints used to predict both good parses and good translations might be pruned in a motivated fashion.

Thesis structure Broadly speaking, this thesis is structured as follows. In chapters 2 and 3, we focus on parsing with the DOP model. We review the work which has been carried out to date for this model in terms of theory, practice and performance. We then discuss in detail how we have opted to implement DOP and the empirical results we have

achieved using our system. In chapters 4, 5, and 6, we investigate the DOT model of translation. We review the theoretical description of the model given by (Poutsma, 1998, 2000, 2003), as well as discussing how it relates to other MT methodologies, the algorithms we have chosen to implement it and the empirical results we have achieved through our implementation. Finally, in chapters 7 and 8, we discuss the data-oriented models of parsing and translation based on the representations of Lexical-Functional Grammar (LFG) in both theoretical and practical terms. The following gives a more detailed description of the material we present.

Chapter 2 Data-Oriented Parsing (DOP) was first introduced in (Scha, 1990; Bod, 1992). In this chapter, we give an overview of the state of the art for this model. Firstly, we focus on the general characteristics of DOP by looking at the types of dependencies captured, and how they differ from those captured by other experience-based parsing methodologies. We then give a more precise description of the DOP model in terms of how fragments – which take the place of rules in a DOP grammar – are induced and how they are used to assign structure to previously-unseen input strings. While the DOP model displays interesting characteristics, empirical evaluation is extremely challenging due to the complexity of both the induced grammars and the required probability model. We discuss pruning techniques which have been proposed to reduce grammar size and how these reductions impact on parse accuracy (e.g. (Bod, 1995b; Sima'an, 1995a; Bod, 2001, 2003b)). We also describe the solutions which have been developed to date to address the tasks of building the DOP parse space for an input string (e.g. (Bod, 1995a; Goodman, 1998; Sima'an, 1999)) and selecting the best parse from that space according to the model (e.g. (Bod, 1995a, 2000e; Chappelier and Rajman, 2003)). Finally, we present alternatives to the DOP fragment probability estimation method (e.g. (Bonnema et al., 2000; Sima'an and Buratto, 2003)) which has been shown to be unsatisfactory (Bonnema et al., 2000; Johnson, 2002).

Chapter 3 In this chapter, we present the DOP system we have developed in terms of implementation and performance. Firstly, we describe the algorithms used to implement each component of our parser. We intend this parser to form the core technology

behind implementations of parsing and translation models which assume tree-based representations encoding more information than simple phrase-structure trees. Consequently, we motivate our choice of algorithm for each task to be achieved in terms of both efficiency *and* flexibility. We then go on to outline the English and French parsing experiments we performed and present a detailed evaluation of the results achieved.

Chapter 4 The main machine translation (MT) paradigms in current use are rule-based MT and data-driven MT. In this chapter, we describe these paradigms and discuss methods of creating hybrid models which embody the positive characteristics of both. We then describe the Data-Oriented Translation (DOT) model of MT (Poutsma, 1998, 2000, 2003) both in general and formal terms. We discuss how DOT relates to both the rule-based and data-driven methodologies and show that it relies on linguistics, statistics and examples to equal degrees.

Chapter 5 Previous empirical evaluation of the DOT model (Poutsma, 2000, 2003) indicated that, despite the attractive characteristics it displays on a theoretical level, the model performs poorly on real data. In this chapter, we assess the reasons for this disappointing performance. Our findings lead us to conclude that a rigorous examination of the performance of the DOT model requires a more robust implementation built to facilitate experiments using larger, more complex datasets than heretofore. Accordingly, in the remainder of this chapter we describe how we have applied the innovative solutions to the challenges of implementing the DOP model in building our DOT system.

Chapter 6 In this chapter, we describe a larger-scale, more informative assessment of the DOT model than before. We describe our experiments in terms of the data used and the evaluation metrics upon which our assessment is based. We then go on to give translation accuracy results over variations in system setup and provide detailed analysis of our findings. Our evaluation shows that the DOT model is capable of generating high-quality translations which are faithful to the data being modelled in terms of both meaning and style.

Chapter 7 The expressive power of the DOP model is limited by the corpus representations it assumes, and phrase-structure trees reflect surface syntactic phenomena only. In this chapter, we describe the DOP model which assumes Lexical-Functional Grammar (LFG) representations developed by Bod and Kaplan (1998, 2003) and summarise the parsing results achieved using this model. We then go on to propose an alternative method of specifying fragment constraints and an alternative method of defining fragment probabilities (Hearne and Sima'an, 2003), and describe the application of efficient DOP algorithms to this model.

Chapter 8 Way (1999, 2001) investigates the possibility of merging the DOT model of translation with LFG representations. In this chapter, we describe the models he proposed and present an alternative model. We discuss how this model might be implemented based on our findings with regard to the implementation of the DOT and LFG-DOP models. We also discuss the implications of moving from phrase-structure trees to LFG representations for the expression of translational equivalence and hypothesise as to how the feature sets for both monolingual and bilingual fragments of LFG representations might be pruned.

Chapter 9 Finally, we conclude and give some avenues for future work.

Chapter 2

The state of the art in Data-Oriented Parsing

Data-Oriented Parsing (DOP) is an experience-based approach to natural language parsing where input sentences are analysed by referencing prior analyses of similar sentences. According to Bod (2003a), collections of analysed sentences were previously used to estimate rule probabilities for hand-written grammars but the DOP model, of which the earliest implementation is described in (Bod, 1992), was the first model to employ prior analyses directly when parsing new input. In section 2.1, we discuss some of the lexical and structural dependencies captured by different approaches to experience-based parsing where context-free phrase-structure tree representations are assumed. We also give a general description of the DOP model – before describing it more formally in section 2.2 – and highlight some dependencies which are captured naturally using DOP.

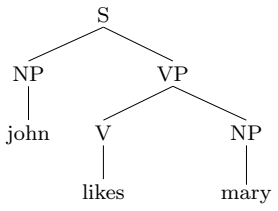
While the DOP model displays properties which are theoretically attractive, researchers interested in submitting the model to empirical evaluation have been faced with serious difficulties. DOP grammars projected from a treebank are large and unwieldy and, in the worst case, exponential in size (relative to the size of the treebank). Consequently, parsing with these grammars is prohibitively expensive in terms of both time and space. Furthermore, calculation of the most probable parse for the DOP model, which is a sum-of-products model, has been shown to be an NP-hard problem (Sima'an, 1995b, 1999, 2003). In sections 2.3–2.5, we describe methodologies which have been developed to address each

of these issues. Finally, it has been shown that using the relative frequency estimator to assign probabilities in the DOP model is unsatisfactory (Bonnema et al., 2000; Johnson, 2002; Sima'an and Buratto, 2003). In section 2.6, we illustrate why this is the case and outline three solutions which seek to address this problem.

2.1 Probabilistic syntax: modelling lexical and structural dependencies

A parser assigns one or more structural analyses to each natural language string it receives as input according to a given grammar if the string is in the language defined by the grammar. A *probabilistic* parser also ranks the analyses assigned to each string, where these rankings are calculated according to weights assigned to each rule in the grammar. It has become usual to extract probabilistic context-free grammars (PCFGs) from treebanks – collections of sentences which have been annotated with context-free phrase-structure tree representations – by extracting the rules which occur in the treebank along with their relative frequencies.

Each PCFG rule extracted from a treebank corresponds to a treebank tree node: the node category appears on the left-hand side of the rule and each of its child node categories or terminal symbols (i.e. words) appears on the right-hand side. The probability attached to any rule in the grammar is its frequency in the treebank conditioned on its left hand side i.e. $P(L \longrightarrow R_1 ... R_x) = \frac{|L \longrightarrow R_1 ... R_x|}{|L \longrightarrow *|}$. Example (2.1) shows a treebank tree and the CFG rules which can be extracted from it, along with their relative frequencies.



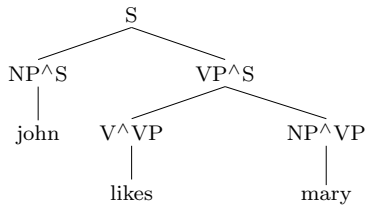
(i)	S	$\longrightarrow$	NP	VP	(1)
(ii)	VP	$\longrightarrow$	V	NP	(1)
(iii)	V	$\longrightarrow$	likes		(1)
(iv)	NP	$\longrightarrow$	john		$(\frac{1}{2})$
(v)	NP	$\longrightarrow$	mary		$(\frac{1}{2})$

(2.1)

The disappointing performance of parsers trained on such PCFGs is mainly attributed to the fact that they do not model non-local dependencies as rule applications are assumed to be independent. The probability of choosing a rule at any step in a derivation does

not reflect the derivation steps seen previously, meaning, for example, that simple PCFGs do not distinguish probabilistically between noun phrases occurring in subject and object positions. This is illustrated in example (2.1) where rules (iv) and (v) are equally likely to be applied when expanding a subject NP despite the fact that, given the evidence in the treebank, *john* is far more likely to appear in subject position than *mary*. Similarly, such PCFGs do not distinguish probabilistically between verbs with different subcategorisation requirements. For example, there is no explicit connection between the surface form of the transitive verb *likes* and rule (ii) which facilitates the analysis of such transitive verbs; if the grammar also contained the rule VP $\longrightarrow$ V, this V could legitimately be expanded using rule (iii) to generate an (incorrect) intransitive reading for *likes*.

Techniques have been developed which go some way towards relaxing these independence assumptions and, consequently, significantly improving PCFG performance. These techniques generally involve transforming the treebank trees in some way and then extracting the PCFG from this transformed treebank. One such transformation, dubbed 'parent annotation' and investigated by Johnson (1999), appends the category of each parent node to the category labels of all its non-terminal children. This process is illustrated in example (2.2), where each node in the treebank tree shown in example (2.1) has been annotated with its parent category and the corresponding rules and probabilities extracted.



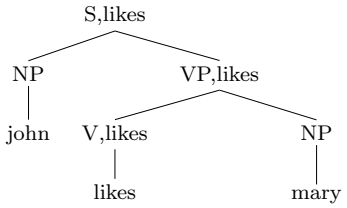
(i)	S	$\longrightarrow$	NP^S	VP^S	(1)
(ii)	VP^S	$\longrightarrow$	V^VP	NP^VP	(1)
(iii)	V^VP	$\longrightarrow$	likes		(1)
(iv)	NP^S	$\longrightarrow$	john		(1)
(v)	NP^VP	$\longrightarrow$	mary		(1)

(2.2)

This parent-annotated PCFG explicitly captures the distinction between the NP occurring in subject position, which is marked with the category NP^S and is used in rules (i) and (iv), and the NP occurring in object position, marked with the category NP^VP and used in rules (ii) and (v).

Another treebank transformation, called 'head-lexicalisation' and introduced by Carroll and Rooth (1998), projects head words up chains of categories by appending them

to the label of each phrase of which they are head. This transformation is illustrated in example (2.3) where the V, VP and S nodes have been annotated with their head surface form *likes*.



(i)	S,likes	$\longrightarrow$	NP	VP,likes	(1)
(ii)	VP,likes	$\longrightarrow$	V,likes	NP	(1)
(iii)	V,likes	$\longrightarrow$	likes		(1)
(iv)	NP	$\longrightarrow$	john		$(\frac{1}{2})$
(v)	NP	$\longrightarrow$	mary		$(\frac{1}{2})$

(2.3)

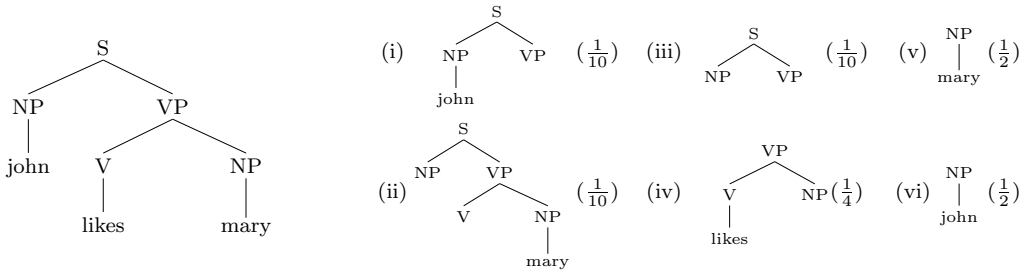
Subcategorisation information for the verb *likes* is made explicit by this head-lexicalised PCFG in rule (ii) as it states that *likes* is followed by an object NP.

Both the Data-Oriented Parsing (DOP) (e.g. (Bod, 1998; Bod et al., 2003)) and PCFG approaches to syntactic parsing (e.g. (Johnson, 1999)) are experience-based in that they learn by extrapolating syntactic generalisations, along with their probabilities, from a set of example parses. The DOP methodology is applied in precisely the same way as the PCFG methodology described in the preceding paragraphs: a probabilistic grammar is extracted from a treebank and used to parse previously unseen input. However, DOP and PCFG models differ in terms of the types of generalisations (i.e. grammar rules) extracted from the treebank and, consequently, how the output parses are ranked.

A DOP grammar comprises tree fragments which are extracted by breaking treebank trees into smaller parts and recombined using a substitution operation to parse new input. Example (2.4) gives just some[1] of the fragments (i) – (vi), along with their relative fre-

[1] A comprehensive specification as to (i) what constitutes a valid DOP fragment and (ii) how the set of valid DOP fragments is extracted from a given treebank tree is given in section 2.2.2; an example illustrating this process is given in Figure 2.3.

quencies,[2] which are extracted from the example treebank tree under the DOP approach.



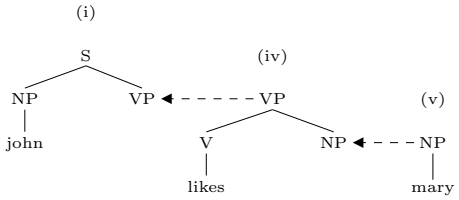
(2.4)

Without making any alterations to the treebank tree, the DOP fragments also model those dependencies captured by the parent and head-lexicalisation transformations. For example, fragments (i) and (ii) explicitly express the distinction between NPs likely to occur in subject position (*john* in this case) and those likely to occur in object position respectively. Similarly, fragment (iv) encodes subcategorisation information about the verb *likes*, indicating that it takes a direct object.

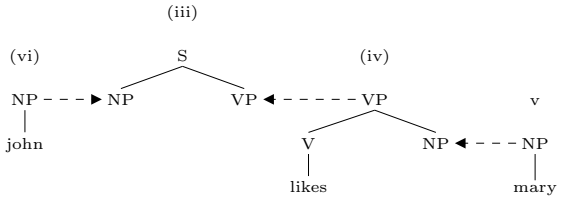
The DOP grammar contains all possible fragments which can be extracted from the treebank trees. This means that, in contrast to the transformed rule sets, the set of DOP fragments also includes all the PCFG rules which can be extracted from the (unannotated) treebank tree (i.e. those in example (2.1)) as depth-1 fragments (e.g. fragments (iii), (v) and (vi) in example (2.4)). Clearly, all fragments in the fragment base overlap to a certain extent: example (2.4) shows that fragment (v) overlaps with fragment (ii), fragment (vi) with fragment (i) and fragment (iii) with both fragments (i) and (ii). Consequently, many different combinations of fragments can yield exactly the same analysis for a given input string. For example, fragments (i), (iv) and (v) can be combined as shown in example (2.5) to form a parse for the string *john likes mary*, as can fragments (iii), (vi), (iv) and (v) as shown in example (2.6). Each of these fragment combinations is viewed probabilistically as a piece of evidence in favour of that parse tree and, thus, plays a part in determining

[2]The 6 example fragments shown here constitute a subset of the set of 17 fragments which can be extracted and the relative frequencies given are calculated over the full fragment set: 10 fragments have root node S, 4 have root node VP, 2 have root node NP and 1 has root node V.

the final ranking of the output parses.



(2.5)



(2.6)

This characteristic of DOP grammars – whereby the fragments are generalised to varying degrees from those which incorporate wider context to express particular dependencies such as (ii) to those which are very general and correspond to plain PCFG rules such as (iii) – is not explicitly shared by PCFGs. Consequently, PCFGs extracted from transformed treebanks tend to suffer from greater sparse data problems. For example, the DOP fragment set in example (2.4) and the PCFG rules in examples (2.1) and (2.3) can be used to parse the sentence *mary likes john*. However, this sentence can no longer be parsed using the parent-annotated PCFG in example (2.2). Similarly, while fragment (iii) in the DOP fragment set and rule (i) in the plain and parent-annotated PCFG rule sets can all potentially be used to parse sentences where a verb other than *likes* is preceded by an NP, this is not the case for the head-lexicalised rules as they specify that the surface form of the head verb must be *likes*.[3]

As demonstrated by the simple examples in (2.1) – (2.3), transformation techniques successfully weaken some of the independence assumptions inherent in PCFGs, thereby inducing more fine-grained models which reflect certain functional and subcategorisation

[3]The effects of the sparse data problem in transformed PCFGs can be addressed to a certain extent by using larger treebanks, and are generally further counteracted through the use of sophisticated smoothing and back-off techniques. The simple grammar transformation examples given here are designed to illustrate the type of dependencies captured by such models; a full exposition of the topic is beyond the scope of this thesis. State-of-the-art PCFG parsers which utilise these techniques are of far greater complexity than these examples would suggest; this is also true of DOP parsers, as will be shown in sections 2.3 – 2.5 of this chapter.

preferences. These techniques can also be successfully applied when modelling real treebanks. For example, the dependency between *connects* and *to* in Figure 2.1(a) can be captured by the rule in (2.7) if the tree is head-lexicalised.

$$VP, connects \longrightarrow V, connects \quad NP \quad PP, to \tag{2.7}$$

In addition, the fact that the NP *the HomeCentre* in Figure 2.1(a) is functioning as direct object while the NP *the PC* is an oblique object can be reflected in the model by applying the parent annotation transformation, which marks the leftmost NP as the child of a VP (NP^VP) and the rightmost as the child of a PP (NP^PP).

It is also the case, however, that these transformations cannot capture *all* relevant dependencies, whether structural or lexical. In Figure 2.1(b), for example, *from* and *to* remain probabilistically independent despite the fact that they are both head words because the intervening NPadj is headed by the noun *page*. Furthermore, the relationships between *last* and *first* and *from last N to first N* are not modelled as neither *last* nor *first* are head words. Furthermore, expressions such as *keep an eye on* in Figure 2.1(c) are modeled solely as the sum of their parts; the idiomatic nature of the expression is not recognised. In contrast, the DOP approach allows the extraction of fragments which express these dependencies also. The partial trees in Figure 2.2 are examples of such fragments: fragment 2.2(a) expresses explicitly the relationship between *connects* and *to* and the functions of the leftmost and rightmost NPs (as direct object and oblique object), fragment 2.2(b) models *from last N to first N* and the idiomatic expression *keep an eye on* is captured in fragment 2.2(c). Note that it is frequently the case that these analyses, along with other analyses for the same strings, can be generated by combining more generalised fragments. Each of the fragments in Figure 2.2, however, strengthens the probabilistic case in favour of the analysis it yields.

In summary, DOP fragments provide snapshots of the many lexical and structural dependencies present in a given treebank. As discussed, some of these distinctions can also be captured via tree transformations used in conjunction with PCFGs. However, DOP fragments also allow us to succinctly capture many more dependencies – such as those discussed in the preceding paragraphs and exemplified by fragments 2.2(b) and (c) –

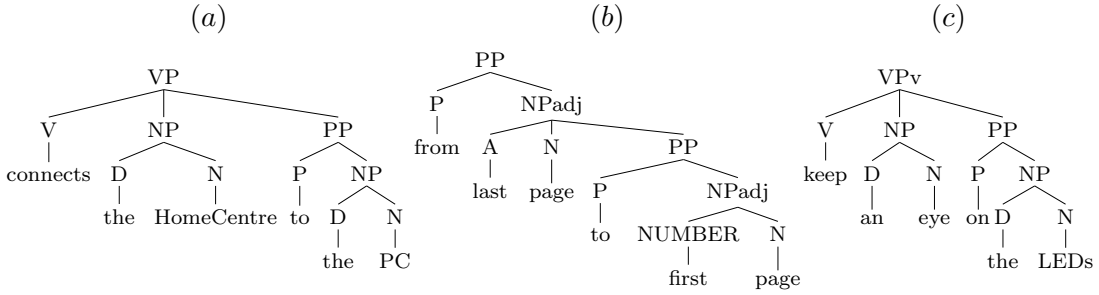


Figure 2.1: Example treebank structures exhibiting dependencies which are difficult to model using PCFGs trained on (transformed) treebanks.

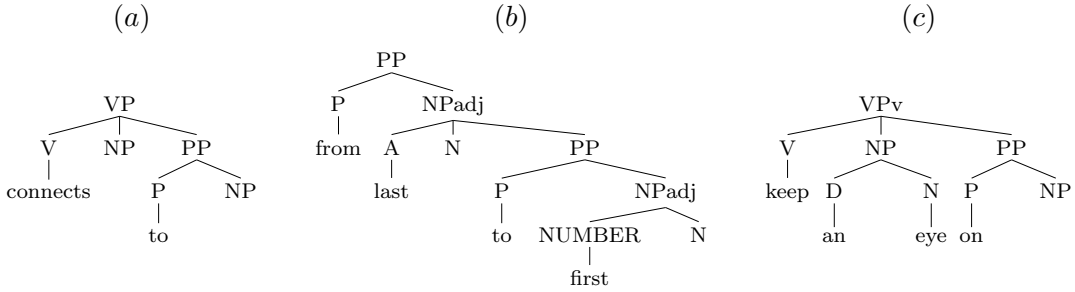


Figure 2.2: Examples of DOP fragments – extracted from the treebank trees in Figure 2.1 – capturing dependencies which are difficult to model using treebank-induced PCFGs.

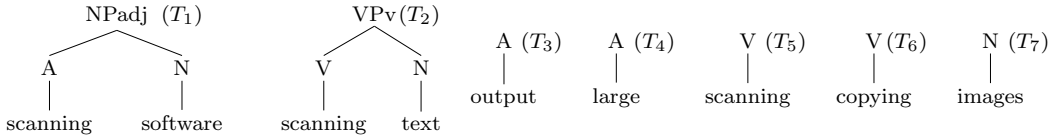
which are far more difficult to capture using PCFGs extracted from plain or transformed treebanks.

In the next section (section 2.2), we provide a formal specification of the basic DOP model and also introduce the stochastic tree-substitution grammar formalism of which DOP is an application (section 2.2.5).

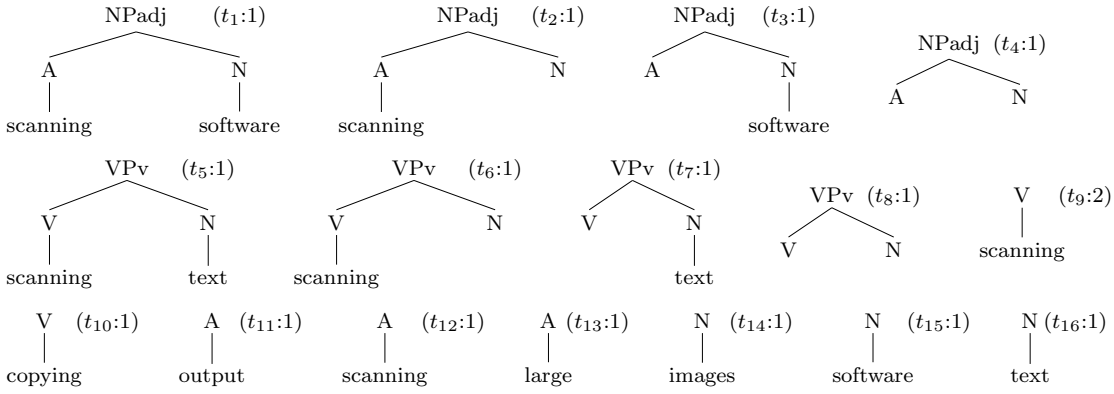
2.2 The Tree-DOP model

In order to describe the DOP model we must specify four elements: the type of representation we expect to find in the example base, how fragments are to be extracted from those representations, how extracted fragments are to be recombined when forming analyses of new input strings, and how the resulting analyses are to be ranked. In the following sections 2.2.1 – 2.2.4, we provide the details of each of these elements for the Tree-DOP model using the illustration provided in Figure 2.3.

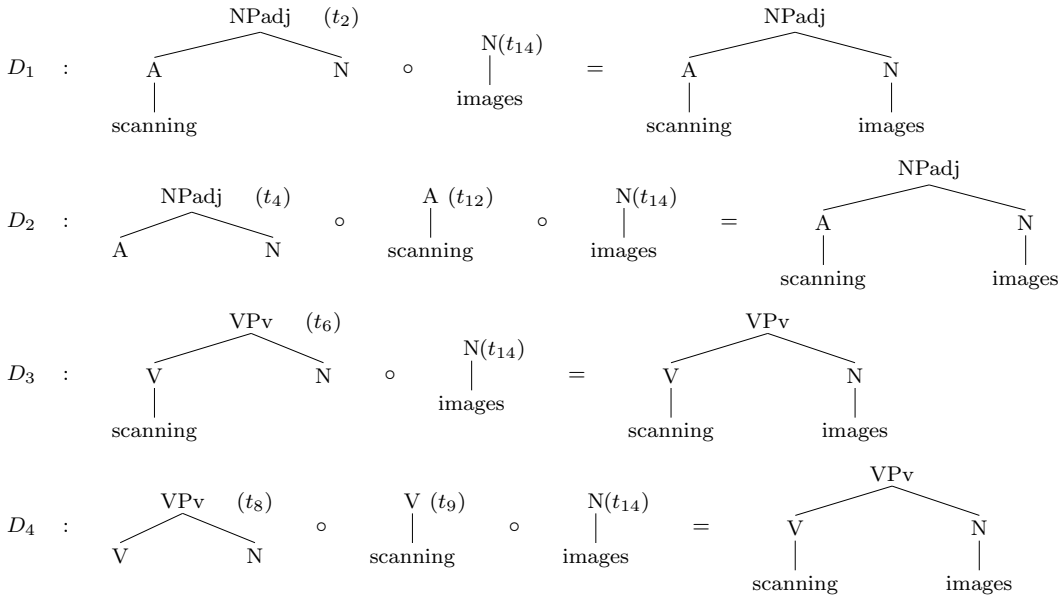
(**A**) **A sample treebank** :



(**B**) **The corresponding fragment set t_1...t_{16} and their associated frequencies** :



(**C**) **All derivations for the input string** *scanning images* **according to fragment base** (**B**) :



(**D**) **Calculation of probabilities corresponding to the derivations in** (**C**) **for** *scanning images* :

$$P(D_1) = P(t_2) * P(t_{14}) = \frac{1}{4} * \frac{1}{4} = \frac{1}{12}$$

$$P(D_2) = P(t_4) * P(t_{12}) * P(t_{14}) = \frac{1}{4} * \frac{1}{3} * \frac{1}{3} = \frac{1}{36}$$

$$P(D_3) = P(t_6) * \mathrm{P}(t_{14}) = \frac{1}{4} * \frac{1}{3} = \frac{1}{12}$$

$$P(D_4) = P(t_8) * P(t_9) * P(t_{14}) = \frac{1}{4} * \frac{2}{3} * \frac{1}{3} = \frac{1}{18}$$

Figure 2.3: Illustration of the process of representation, fragmentation, composition and ranking for Tree-DOP.

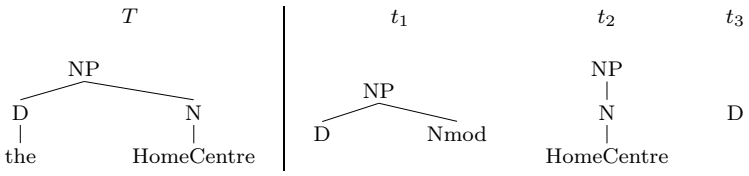


Figure 2.4: Fragments t_1 – t_3, when extracted from tree T, are not valid fragments as they each fail to meet one or more of the criteria specified in section 2.2.2.

2.2.1 Representations

Many different linguistic formalisms can be used to annotate the database of examples which underpins all DOP systems. In this section, we assume the representations used when developing the earliest (and most common) DOP systems: syntactically labelled context-free phrase structure trees (henceforth "trees"). A sample treebank is given in Figure 2.3(A).

2.2.2 Fragmentation

The fragmentation process involves extracting generalised fragments (or subtrees) from the trees contained in the example base. A fragment t extracted from tree T is valid only if it meets the following criteria:

1. each node in t is a node in T,
2. each node in t either has no children or has exactly the same number of children as the corresponding node in T, and
3. t consists of more than one node.

Examples of fragments which do not meet these criteria given tree T are given in Figure 2.4: t_1 violates (1) as the node Nmod does not appear in tree T, t_2 violates (2) as it indicates that the NP node has only one child (N) whereas in T the NP node has two children (D and N), and t_3 violates (3) as it is comprised of just one node.

Fragments are systematically extracted from the annotated example base using the *root* and *frontier* operations. These are defined as follows (Bod, 1998):

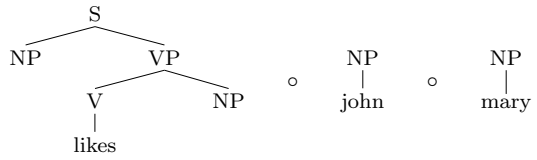
- given a copy of tree T called T_{copy}, select a node to be *root* and delete all except the subtree it dominates, and
- select a set of nodes in T_{copy} to be *frontier* nodes and delete the subtrees they dominate.

For example, fragment t_4 in Figure 2.3(B) was extracted by making a copy of tree T_1 from treebank (A) called t_4, selecting the node labelled NPadj to be root, adding the nodes labelled A and N to the frontier set and deleting the subtrees they dominate. Similarly, fragment t_{16} was extracted by making a copy of tree T_2, selecting the node labelled N to be root, deleting all subtrees except the subtree it dominates and selecting the empty set as the frontier set.

2.2.3 Composition

Each fragment frontier (or leaf) node is either a terminal symbol (i.e. a word) or a syntactic category. Frontier nodes which are syntactic categories constitute open substitution sites; fragments whose root node syntactic category matches that of the frontier node can be substituted at that frontier node. For example, fragment t_2 in Figure 2.3(B) has two frontier nodes, one of which is a terminal symbol labelled *scanning* and the other an open substitution site labelled *N*; any fragment in the fragment base with root node N can be substituted at this site by simply replacing N with the given fragment.

Tree-DOP substitution is achieved via the composition operation ($\circ$). This is a left-most substitution operation, meaning that where a fragment has more than one open substitution site, composition must take place at the leftmost site. This ensures that each derivation is unique. For example, if the composition operation did not specify order then the composition sequence in example (2.8) would have two realisations, one where *john* is in subject position and the other where *john* is in object position. However, as we are required to always compose at the *leftmost* available site, this sequence can actually only realise the parse where *john* is subject.



(2.8)

Tree-DOP derivations are built using the composition operation. Once an initial fragment is chosen to start the derivation, further fragments are successively substituted at the leftmost open substitution site until no open substitution sites remain. Example derivations are given in Figure 2.3(C).

2.2.4 The probability model

Frequently, multiple analyses are assigned to an input string. For example, in Figure 2.3(C) we see that two distinct parses are generated for the string *scanning images*, as the phrase can be analysed both as a noun phrase and as a verb phrase. The probabilities associated with each parse are calculated and used to rank the set of parses and (as is generally required) discern which is the most likely analysis of the input string given the treebank.

Firstly, the probability of a fragment f_x with root node X is its relative frequency in the set of fragments with root node X, as given in (2.9):[4]

$$P(f_x) = \frac{|f_x|}{\sum_{root(f)=root(f_x)} |f|} \tag{2.9}$$

The probability of each derivation D_x is then defined as the product of the probabilities of the fragments used to build that derivation, as given in (2.10):

$$P(D_x) = \prod_{f \in D_x} P(f) \tag{2.10}$$

Finally, the probability of a parse T_x is the sum of the probabilities of the derivations that yield that parse, as given in (2.11):

$$P(T_x) = \sum_{D \; yields \; T_x} P(D) \tag{2.11}$$

[4]The majority of DOP implementations to date have estimated fragment probabilities according to their relative frequencies; alternative methods of accomplishing this task are discussed in section 2.6.

The calculation of derivation probabilities is illustrated in Figure 2.3(D). Note that the highest derivation probability is $\frac{1}{12}$ and is assigned to two derivations, D_1 and D_3. In this example, therefore, calculating the most probable derivation does not allow us to distinguish which parse tree is the 'best' parse for the given input string. However, calculation of parse probabilities, which requires us to sum over derivation probabilities, results in just one most probable analysis: the tree yielded by derivations D_3 and D_4 has probability $\frac{5}{36}$ whereas the tree yielded by derivations D_1 and D_2 has a lower probability $\frac{4}{36}$. This is due to the fact that there are two instances of *scanning* as a verb in the treebank (out of a total of three instances of verbs) whereas there is only one instance of *scanning* as an adjective (also out of a total of three adjectives).

2.2.5 Tree-DOP as a Stochastic Tree-Substitution Grammar

The Tree-DOP model can be viewed as an instantiation of a Stochastic Tree-Substitution Grammar (STSG) (Bod, 1998). As DOP grammars are frequently referred to as STSGs in the literature (e.g. (Sima'an, 1995a, 1999; Chappelier and Rajman, 2003)) and the terms are used interchangeably throughout this thesis, we give the formal definition of an STSG here.

A stochastic tree-substitution grammar G is a 5-tuple $< V_N, V_T, S, R, P >$ where

- V_N is a finite set of non-terminal symbols,
- V_T is a finite set of terminal symbols,
- $S \in V_N$ is the start symbol,
- R is a finite set of elementary trees whose root and internal node symbols are elements of V_N and whose leaf node symbols are elements of V_N or V_T, and
- P is a function which assigns a probability $P(t)$ to every $t \in R$ such that $0 \leq P(t) \leq 1$ and $\sum_{t:root(t)=X} P(T) = 1$.

Elementary trees are composed using the leftmost substitution function $\circ$: if t_1,$t_2 \in R$ and the root node symbol of t_2 is the same as the leftmost non-terminal leaf node of t_1 then $t = t_1 \circ t_2$ is produced by substituting t_2 at the leftmost non-terminal leaf node of t_1.

A leftmost derivation is an n-tuple of elementary trees $< t_1, ..., t_n >$ such that

- $t_1 ... t_n \in R$,
- $root(t_1) = S$, and
- the frontiers of $t_1 \circ ... \circ t_n$ are elements of V_T.

Derivation $< t_1, ..., t_n >$ derives tree T if $t_1 \circ ... \circ t_n = T$ and derives string W if $frontier(t_1 \circ ... \circ t_n) = W$. The probability of derivation $< t_1, ..., t_n >$ is the product of the probabilities of the elementary trees $t_1 ... t_n$ used to build it. The probability of a parse is the sum of the probabilities of the derivations which yield that parse. The probability of a string is the sum of the probabilities of the distinct parses which yield that string. The probability of a string is also the sum of the probabilities of its derivations.

Clearly, the Tree-DOP model is an STSG: Tree-DOP fragments correspond to the elementary trees of the STSG and the probabilities of those fragments are the probabilities of the elementary trees.

2.3 Fragmentation in practice

Consider node A_T in treebank tree T which immediately dominates nodes $C_{T_1} ... C_{T_n}$. The number of fragments $F(A_T)$ projected from this node under the DOP model is calculated according to equation (2.12).

$$F(A_T) = (F(C_{T_1}) + 1) * ... * (F(C_{T_n}) + 1) \tag{2.12}$$

The total number of fragments $TF(T)$ which can be extracted from treebank tree T is the sum over the number of fragments which can be projected from each of its nodes, as stated in equation (2.13).

$$TF(T) = \sum_{A_T \in T} F(A_T) \tag{2.13}$$

Applying equation (2.13) to the leftmost tree in Figure 2.5, for example, indicates that it yields 87 fragments.

The set of fragments projected from a treebank of even relatively modest size is generally extremely large. For example, the English section of the HomeCentre corpus,[5] which contains 980 trees, yields 308,486,334,496 DOP fragments. Many different heuristics have been used to limit the size of the fragment base. These generally involve setting upper limits on fragment characteristics such as depth, number of lexical items, number of open substitution sites and combinations thereof; fragments which exceed these limits are excluded from the fragment base.

The aim in introducing such heuristics is to reduce the number of fragments in the training set. Consider, for example, the trees in Figure 2.5 which have all been extracted from the tree to the left of the vertical rule using the *root* operation. These are intermediate trees, i.e. in order to extract DOP fragments, the *frontier* operation is applied one or more times to each of these trees in turn. Restricting the set of fragments extracted with respect to **depth** – where tree depth is the longest length of the path from the root node to a frontier node – involves placing conditions on the sets of nodes selected by the frontier operation each time it is applied. If we set the maximum fragment depth to 3, then the set of frontier nodes applied to the intermediate tree with root node C in Figure 2.5 must contain either nodes H and I or a non-root node which dominates H and I, i.e. E or G. In other words, any fragment of depth 3 extracted from the tree with root C must have nodes H and I as open substitution sites; fragments to which H and I are internal exceed the maximum depth and are discarded. As previously stated, the maximum number of fragments which can be extracted from the leftmost tree in Figure 2.5 is 87. If, however, we impose an upper limit of 3 on fragment depth then the number of fragments extracted is reduced to 39.

While introducing pruning heuristics has the desired effect of reducing the size of the DOP fragment set, discarding fragments also results in reduced sensitivity to lexical and structural dependencies. For example, excluding those fragments of depth greater than 3 in the example given in Figure 2.5 means that the relationship between the lexical items b and f is not explicitly captured. An important motivation behind the development of the DOP model is that, in contrast to many other experience-based approaches to parsing,

[5]The HomeCentre corpus is described in detail in sections 3.2.1 and 6.1.

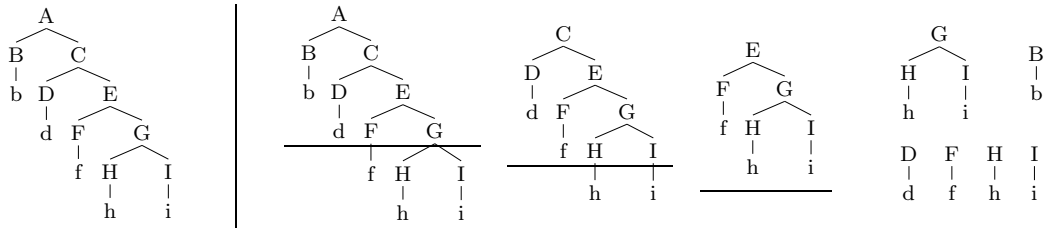


Figure 2.5: The trees on the right were extracted from the leftmost tree via the *root* operation; the *frontier* operation must now be applied (one or more times) to each of these trees in order to extract DOP fragments. If fragment depth is limited to 3 then nodes appearing below the horizontal lines cannot occur in extracted DOP fragments.

it models *all* lexical and structural relationships which occur in the given treebank. Of course, probabilistically modelling many of these perceived relationships may not significantly increase parse accuracy, as we do not know which fragments model relatively weak lexical and structural relationships that do not play a particularly important probabilistic role, and which fragments model strong dependencies which constitute valuable pieces of probabilistic evidence. As the sets of fragments to be excluded are defined in quantitative rather than linguistic terms, the pruning process is also blind to the probabilistic importance of the fragments kept and discarded. In other words, returning to the example in Figure 2.5, as the strength of the relationship between the lexical items b and f is not known, the impact on parse accuracy of *not* modelling this relationship is also unknown.[6]

Clearly, there is strong motivation for using pruning techniques when parsing with the DOP model: even relatively small treebanks yield extremely large numbers of fragments. However, it is necessary to determine which heuristics engender least deterioration in the quality of the output parses. In this section, we describe the heuristics which have been implemented to date to limit grammar size and report on whether or not use of these heuristics has led to reduced parse accuracy.

An empirical study of several of the constraints that can be imposed on the subtrees to be included in the fragment base was performed by Bod (2001, 2003b). Experiments

[6]Of course, aside from the effect on parse quality of not modelling arbitrary dependencies, excluding large numbers of fragments also distorts the frequency distribution of the remaining fragments. As fragment probabilities are estimated according to their relative frequencies in the fragment base, distortion of the fragment distribution has a profound effect on the probability models which result. This issue is discussed in greater detail in section 2.6.

were performed using sections 2–21 of the WSJ Penn-II treebank (Marcus et al., 1994) for training, section 22 for development and section 23 for testing (as standard).[7] As it was not practical (due to memory limitations) to use all possible fragments for training, all fragments of depth 1 and random samples of 400,000 fragments of depths 2 through 14 were used instead. (No fragments of depth greater than 14 were used.) This resulted in a training set containing 5,217,529 fragments; base-line experiments were performed using this set and then further experiments performed by excluding various fragment types from the set.

The first subtree restriction imposed was on **subtree depth**: experiments were performed where the fragment base contained only fragments of depth N or less such that $1 \leq N \leq 14$. These experiments showed that parse accuracy increased as the size of the fragments included in the fragment base increased; the highest scores were obtained when the full baseline subtree set was used.

The impact of **lexical content** was also assessed by excluding from the baseline fragment set those subtrees whose number of lexicalised frontiers exceeded an upper limit. Thus, the fragment set varied from containing fragments which had maximally one lexicalised frontier to containing all fragments (i.e. removing the upper limit). Results show that accuracy increased initially as lexical content was enlarged but started to decrease when the upper limit exceeded 12.[8]

The importance of **structural dependencies** was examined by excluding unlexicalised subtrees of varying depths while retaining all lexicalised subtrees with up to 12 lexicalised frontiers. These experiments showed that accuracy increases when unlexicalised fragments are included in the fragment base up to depth 6; beyond this, unlexicalised fragments do not appear to contribute to parse accuracy for the WSJ corpus.

Restrictions were also imposed on the number of **non-headwords** in lexicalised fragments in order to investigate the importance of non-headword dependencies. Fragments containing more than a set maximum of non-headwords were excluded from the fragment

[7] The parser used to conduct these experiments did not compute the most probable parse; rather the 1000 most probable derivations were calculated using the Viterbi n-best algorithm, and the output parse selected by summing over the DOP probabilities of those derivations yielding the same trees. This method does not guarantee that the best parse is found. This model is described fully in section 2.5.3.

[8] No explanation for this decrease is given but we suspect that it is related to the parameter estimation issue; again, we discuss this further in section 2.6.

base where this maximum varied from one non-headword to an unrestricted fragment set. Results show that inclusion of non-headword dependencies leads to improved parse accuracy, although these improvements are small: the difference between using no headwords and all headwords is 1.2% for precision and 1% for recall.

Experiments investigating the importance of **low-frequency fragments** were carried out on the ATIS treebank and presented in (Bod, 1999). These experiments demonstrated that low-frequency fragments are useful in determining which parse is most appropriate, and that excluding them results in decreased accuracy. Bod (1995b) also showed that, again on the ATIS treebank, excluding all fragments which occur only once in the fragment base leads to a decrease in parse accuracy of 4%.

Experiments investigating the impact of varying the number of **open substitution sites** in each fragment were carried out on the ATIS treebank and presented in (Sima'an, 1995a).[9] These experiments show that including fragments with maximally one open substitution site gives better performance than including those with maximally two open substitution sites. Furthermore, performance was at least as good as when no restriction was placed on the number of open sites.

Of course, all of these constraints on the fragment base are heuristics, and while some may be shown to give better performance than others there is no guarantee that such constraints will give similar results when parsing over DOP grammars induced from different treebanks. As previously stated, if use of such constraints is necessary then the constraint parameters must be empirically determined during a development phase.

2.4 Parsing methodologies for Tree-DOP

During parsing, the input string is associated with all the possible structures which can be assigned to it according to the given grammar. (This is distinct from the disambiguation stage where one of the possible parses is deemed to be the 'best' parse, cf. section 2.5.) These parses are stored on a chart which is usually referred to as a *parse space* or *parse forest*. Rather than explicitly constructing all of the possible parses, the parse space

[9] These experiments were carried out by searching for the most probable derivation rather than the most probable parse.

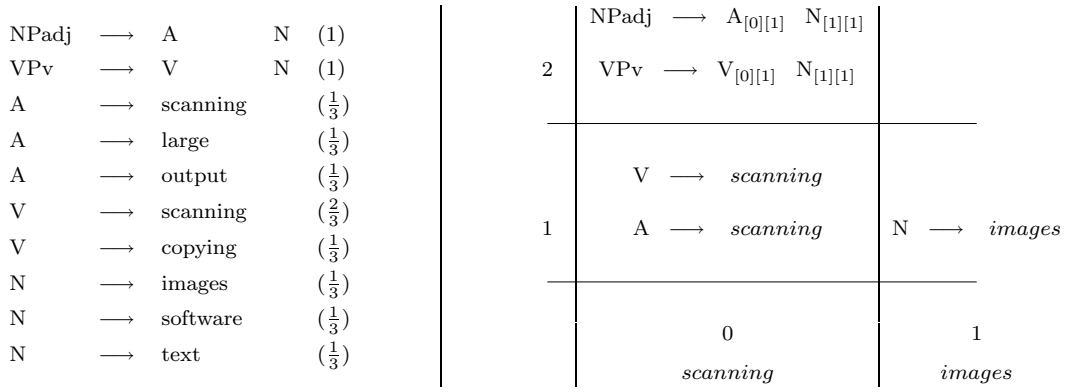


Figure 2.6: The parse space for the input string *scanning images* according to the PCFG on the left extracted from the treebank in Figure 2.3(A).

contains all grammar rules (which, in the DOP model, correspond to fragments) which can be used to parse the current input string, along with pointers to those rules with which they can combine to form valid parses.

Figure 2.6 shows the parse space – a two-dimensional chart of size N^2 where N is the length of the input string – for the phrase *scanning images* according to the PCFG grammar on the left which was extracted from the example treebank given in Figure 2.3(A). Each token in the input string is assigned a number i such that $0 \leq i < N$. These numbers appear along the horizontal axis; the numbers which appear on the vertical axis (generally represented by j) indicate the number of input tokens spanned. Thus, rule r_x appearing in chart position $[i][j]$ signifies that derivations of one or more parse trees representing the portion of the input string which starts with token i and spans j consecutive tokens can be started with rule r_x. Rules in chart position $[0][N]$ span the entire input string; if this position is empty then the input string cannot be parsed. The right-hand side of each PCFG rule can consist of non-terminal and/or terminal symbols. Each non-terminal symbol on the right-hand side of any rule present on the chart can explicitly point to the chart position from which rules which can be combined with it must be selected. For example, the rule with left-hand side NPadj in chart position [0][2] in Figure 2.6 has the non-terminal symbol A on the right-hand side; those rules which can be combined at this position must have left-hand side A and be selected from chart position [0][1].

Standard chart-parsing algorithms that compute the PCFG parse space for a given

input string include the CKY algorithm (Younger, 1967; Aho and Ullman, 1972) and Earley's algorithm (Earley, 1970; Stolcke, 1995).

The CKY algorithm enters rules onto the parse chart in a left-to-right bottom-up manner. This algorithm requires the grammar with which it parses to be in Chomsky-Normal Form (CNF), i.e. the right-hand side of each grammar rule must comprise either one terminal symbol or two non-terminal symbols; converting PCFGs to CNF is straightforward but results in an increase in the size of the grammar. (The magnitude of the increase depends on the lengths of the right-hand sides of the rules in the grammar.) The CKY algorithm comprises a base case and a recursive case. Executing the base case involves filling in the chart entries for row 1, i.e. inserting all rules of the form X $\longrightarrow$ w_i for each word w_i in the input string. Executing the recursive case involves filling in the chart entries for row 2 and upwards. Execution of the recursive case is facilitated by the conversion of the grammar to CNF: all rules which span more than one token have exactly two non-terminal symbols on the right-hand side. At position $[i][j]$, rule X $\longrightarrow$ Y Z can be inserted into the chart if there exists already a rule with left-hand side Y at chart position $[i][k]$ and there exists already a rule with left-hand side Z at position $[i+k][j-k]$ such that $i \leq k \leq j$. Simply noting the chart positions containing rules with which each right-hand side symbol can combine yields a chart such as the one in Figure 2.6.

In contrast, Earley's algorithm is a left-to-right top-down algorithm which can handle rules with an arbitrary number and combination of terminal and non-terminal symbols on their right-hand sides. As the input string $w_1 ... w_n$ is scanned from left to right, a set of states representing each point in the recognition process is constructed. Each state comprises a grammar rule, an indication (taking the form of a dot $\bullet$) as to how much of the right-hand side of that rule has been recognised and an indication as to which w_i caused that rule to be entered. For example, state (i) X $\longrightarrow$ $\bullet$Y Z starts at w_i and none of its right-hand side has yet been recognised; this state in turn causes rules with left-hand side Y to be entered onto the chart. When scanning a word results in full recognition of the right-hand side of a rule (i.e. the chart contains a rule of the form Y $\longrightarrow$ $RHS_1 ... RHS_n \bullet$), this results in incremental recognition of the right-hand sides of further rules on the chart. While this algorithm removes the need to convert rules to CNF, it also inserts many more

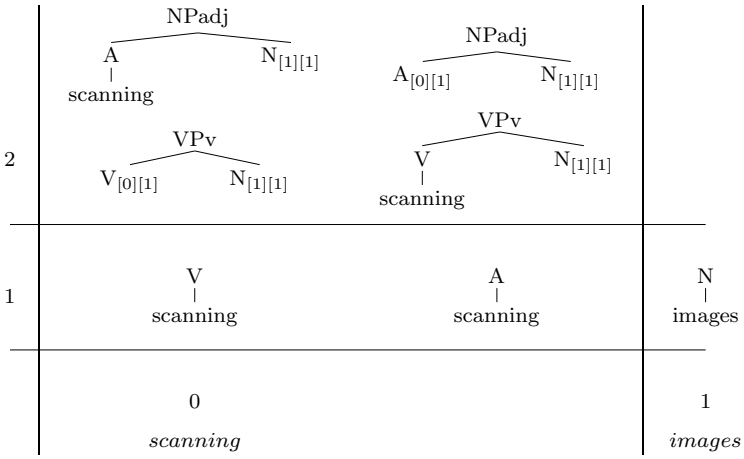


Figure 2.7: The Tree-DOP parse space for the input string *scanning images* given the Tree-DOP grammar in Figure 2.3(B).

failing derivations into the chart and, consequently, is more computationally expensive than the CKY algorithm.

Figure 2.7 also shows the parse space for the input string *scanning images*, this time according to the DOP grammar given in Figure 2.3(B). The first implementation of DOP, described in (Bod, 1995a), used a standard chart-parsing algorithm to accomplish parse-space computation, although precisely which algorithm was used is not specified. Standard algorithms such as CKY and Earley work well with PCFG grammars comprising, perhaps, 50,000 rules. However, even when a DOP grammar has been pruned using the techniques outlined in section 2.3, it is generally the case that the remaining fragment set is still extremely large. Even small DOP grammars are far larger than most PCFGs and, consequently, directly applying standard algorithms is extremely inefficient. In the following sections 2.4.1 – 2.4.3, we give further details of Bod's implementation and outline two alternative parsing algorithms which allow crucial savings in terms of both time and memory.

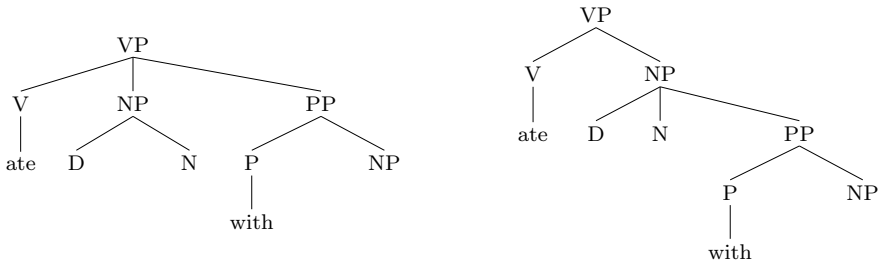
2.4.1 Bod: fragments as re-write rules

The first DOP implementation methodology, introduced in (Bod, 1992) and described in greater detail in (Bod, 1995a), views each tree as a rewrite rule and uses standard chart-parsing techniques to build the parse space for any given input string. Each fragment f

with n frontiers yields a rule of the form (2.14):

$$root(f) \longrightarrow frontier_1(f) \ldots frontier_n(f) \tag{2.14}$$

As many different internal structures can be associated with fragments which have the same root and frontier nodes, each rule must also be associated with the fragment which yielded it. Consider, for example, the fragments in (2.15).



(2.15)

These are both associated with the same rewrite rule *VP* $\longrightarrow$ *ate D N with NP* but the distinction between them is maintained by retaining links to the distinct structures (and their probabilities) with which they are associated.

These rules are then applied to the input string using a standard chart-parsing algorithm; this process results in the parse space for the input string. As each rule references the structure it represents, every unique fragment corresponds to a rule. Thus, if the DOP grammar comprises N fragment types, there will be N corresponding rule types. Given a treebank of reasonable size, the DOP grammar extracted will generally be far too unwieldy for this approach to be practical. If the CKY algorithm is used, each rule must be converted to CNF. As the right-hand side of each rule comprises a sequence of terminal and non-terminal symbols whose maximum length is that of the longest sentence in the treebank, conversion to CNF results in an explosion in the size of the rule set. On the other hand, if Earley's algorithm (even augmented with look-ahead) is used, then a crippling number of failing derivations will inevitably be present in the parse chart. Clearly, alternative techniques are required.

2.4.2 Sima'an: two-phase parsing

Sima'an (1995a, 1999) describes a two-phase parsing methodology for efficiently computing

the DOP parse space of a given input string. The first phase spans a good approximation of the parse space using the (non-probabilistic) CFG underlying the DOP grammar (i.e. the CFG which can be extracted from the treebank, in which all trees are assumed to be binary branching).[10] The second phase then uses correspondences between the CFG rules and the fragments in which they occur to reduce from the CFG parse space to the DOP parse space. Crucial to the effectiveness of this approach is the fact that the CFG underlying a typical DOP grammar is far smaller than the fragment set. The following property of STSGs is essential to this method:

> The string/tree language of an STSG is always a subset of the string/tree language of the CFG underlying it.

In other words, when we approximate the parse space using the underlying CFG in the first phase, we are certain that we have included all parses which can be assigned by the STSG to the input string. This is illustrated in Figures 2.6 and 2.7, where both the CFG parse space and the DOP parse space yield the same parse trees – the fact that the probabilities with which they are produced differ is not relevant to Sima'an's algorithm.

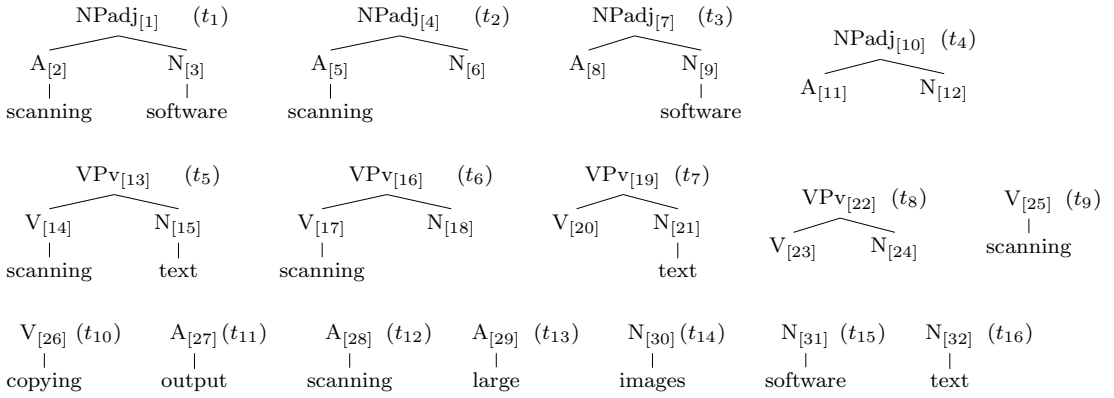
Sima'an (1999):118 specifies the two phases of the algorithm as follows:

> **Phase 1.** Apply the CKY algorithm using the CFG G_{cfg} underlying the DOP grammar G_{stsg}; this yields a parse space which is a superset of that yielded by G_{stsg} for the same sentence.
>
> **Phase 2.** Apply an algorithm to compute the parse space yielded by G_{stsg} *from the approximated parse space yielded by* G_{cfg} and compute the Most Probable Derivation (MPD) on this parse space.

During the second phase, the atomicity and uniqueness of the fragments from G_{stsg} must be preserved and exactly the parse space and derivations of the input string of G_{stsg} must be recognised. In order to fulfil these conditions, a mapping is drawn between the CFG rules and the DOP fragments. Firstly, every node in every fragment in G_{stsg} is assigned a unique address – this is illustrated in Figure 2.8(A) where each node address

[10] The handling of trees which are not binary branching is discussed in section 2.4.4.

(A) A DOP grammar where each node has been annotated with a unique identifier :

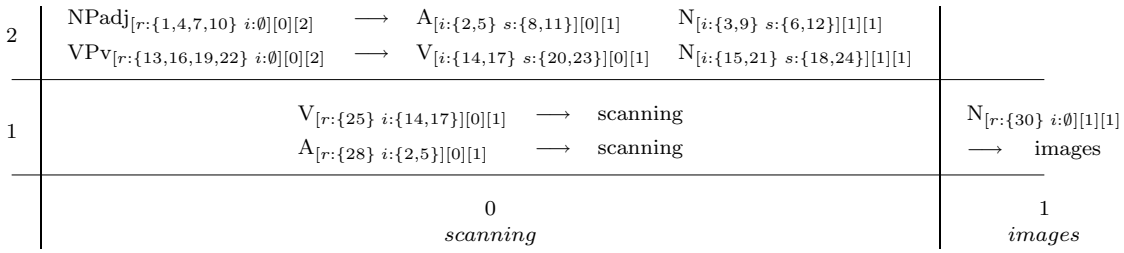


(B) The annotated PCFG underlying the DOP grammar in (A) :

$$\text{NPadj}_{[r:\{1,4,7,10\}\ i:\emptyset]} \longrightarrow \text{A}_{[i:\{2,5\}\ s:\{8,11\}]}\ \text{N}_{[i:\{3,9\}\ s:\{6,12\}]}$$
$$\text{VPv}_{[r:\{13,16,19,22\}\ i:\emptyset]} \longrightarrow \text{V}\ [i:_{\{14,17\}\ s:\{20,23\}]}\ \text{N}_{[i:\{15,21\}\ s:\{18,24\}]}$$

$\text{V}_{[r:\{25\}\ i:\{14,17\}]} \longrightarrow$ scanning
$\text{A}_{[r:\{27\}\ i:\emptyset]} \longrightarrow$ output
$\text{A}_{[r:\{29\}\ i:\emptyset]} \longrightarrow$ large
$\text{N}_{[r:\{31\}\ i:\{3,9\}]} \longrightarrow$ software
$\text{V}_{[r:\{26\}\ i:\emptyset]} \longrightarrow$ copying
$\text{A}_{[r:\{28\}\ i:\{2,5\}]} \longrightarrow$ scanning
$\text{N}_{[r:\{30\}\ i:\emptyset]} \longrightarrow$ images
$\text{N}_{[r:\{32\}\ i:\{15,21\}]} \longrightarrow$ tabular

(C) The annotated PCFG parse space according to the grammar in (B) :



(D) The DOP parse space calculated from the PCFG space in (C) using the viability property :

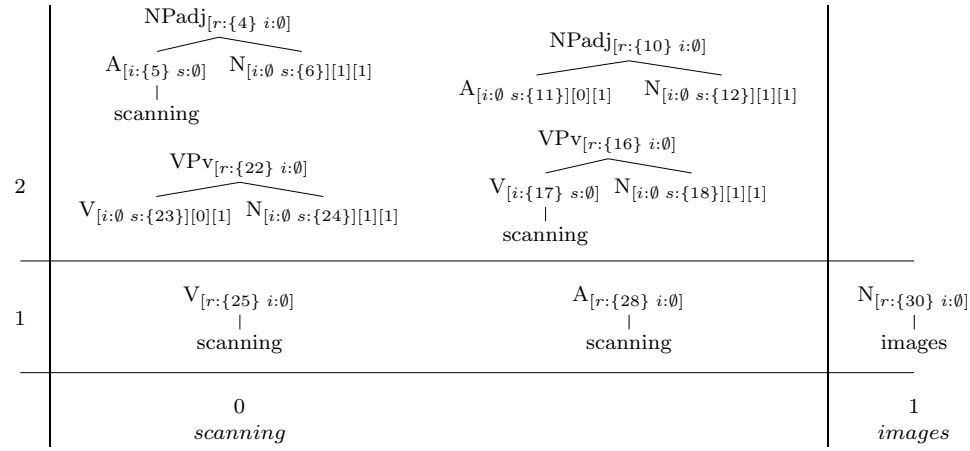
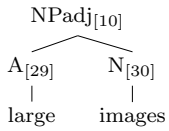


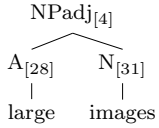
Figure 2.8: Calculation of the DOP parse space for the string *scanning images* using Sima'an (1999)'s two-phase analysis algorithm.

is an integer displayed in square brackets. Every derivation (i.e. sequence of fragment compositions yielding a tree containing no open substitution sites) which can be generated by G_{stsg} is thus characterised by the unique addresses assigned to its nodes. For example, the addresses assigned to the nodes of example tree (2.16) indicate that it must have been derived by the composition sequence $(t_4 \circ t_{13} \circ t_{14})$ and no other.



(2.16)

Each rule in G_{cfg} is associated with the set of addresses of all nodes to which it corresponds as illustrated in Figure 2.8(B).[11] The first parsing phase uses this annotated PCFG to generate an annotated G_{cfg} parse space such as the one in Figure 2.8(C). Each derivation in this G_{cfg} parse space is associated with one or more sets of node address assignments. However, not every set of node address assignments associated with each derivation corresponds to a G_{stsg} derivation. This is illustrated in example (2.17): the left child of node $\text{NPadj}_{[4]}$ must be an internal node but according to this derivation it is an open substitution site at which any fragment whose root node category is A can be inserted.



(2.17)

Thus, Sima'an (1999):119 defines a procedure which recognises whether or not a set of node address assignments is a valid G_{stsg} derivation using the *viability property*:

> **Viability property.** T is a tree derived from G_{cfg} which is associated with one or more sets of node address assignments. T' is an instance of T decorated with just one set of these addresses. X is a node in T' with label N and address c. It has a j^{th} child Xch labelled N_j with address c_j. The viability property for X and Xch holds if one of the following holds:

[11] In the annotated PCFG in Figure 2.8(B), we have partitioned the sets of addresses according to whether they correspond to root nodes (r), internal nodes (i) or substitution sites (s). The categories on the left-hand side of each rule cannot correspond to internal nodes and the categories on the right-hand side of each rule cannot correspond to root nodes. Although partitioned here for the sake of clarity, partitioning of address sets also results in greater efficiency and partitions based on other criteria are also possible.

Parenthood. N's code c and N_j's code c_j correspond to a parent and its j^{th} child in some fragment; note that one unique address c_j can be the j^{th} child of N at c.

Substitution. N's code c appears in a fragment with an open substitution site N_j as its j^{th} child and N_j's code c_j corresponds to the root of a fragment labelled N_j.

Accordingly, during Phase 2 each set of node address assignments corresponding to a particular G_{cfg} tree derivation is examined. Each node address must correspond to either a root node or an internal node; valid derived trees comprise groups of internal nodes delimited by root nodes (each of which, apart from the root node of the derivation, is also an open substitution site) where each group corresponds to a G_{stsg} fragment. It is useful to construct node addresses such that the parent-child and sisterhood relations can be efficiently checked. It is also useful to partition the sets of node addresses corresponding to particular G_{cfg} rules according, as illustrated, to whether or not a particular address is a root node and whether or not it has children; such encodings results in further speed-ups.

Sima'an's approach allows computation of the DOP parse space without looking back to the original DOP grammar as the relevant fragments are essentially reconstructed using the node address annotations on the CFG rules. However, it does not allow us to compute fragment probabilities directly; these must be retrieved from the original grammar by using the node addresses to identify the corresponding fragments. Sima'an continues by outlining a method to compute the MPD but his optimised DOP parsing algorithm can be also used in conjunction with other disambiguation strategies.

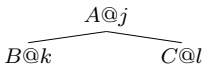
2.4.3 Goodman: parsing with PCFG-reductions

Goodman (1996a, 1998, 2003) describes a method by which the DOP grammar projected from a treebank in which all trees are binary branching[12] is reduced to a PCFG containing at most eight rules for each node in the training data. This PCFG is equivalent to the DOP grammar in that a) it generates the same strings with the same probabilities and

[12] Again, the handling of trees which are not binary branching is discussed in section 2.4.4.

b) it generates the same parse trees with the same probabilities, although one must sum over several PCFG trees for each DOP tree.

Goodman PCFG-reductions are constructed as follows. Every node in every tree in the treebank is assigned a unique address: $A@k$ is the node labelled A at address k. One new non-terminal A_k is created for every node in the treebank; such non-terminals are called "interior" nodes and the original nodes "exterior" nodes. a_k is the number of subtrees with root node $A@k$ and a the number of subtrees with root node label A, i.e. $a = \sum_j a_j$. Given node $A@k$ with a set CH of two or more children $CH = \{B@l...C@m\}$, the number of fragments a_k which have root node $A@k$ is calculated by multiplying the numbers of fragments which each of its child nodes yields: $a_k = \prod_{X@n \in CH}(x_n + 1)$.



(2.18)

For any node grouping such as the one in example (2.18), the eight PCFG rules and their corresponding probabilities in example (2.19) are then extracted; Goodman provides proofs by induction that the rule probabilities are valid.

$$
\begin{array}{llllllllll}
(1) & A_j & \longrightarrow & BC & (\frac{1}{a_j}) & (5) & A & \longrightarrow & BC & (\frac{1}{a}) \\
(2) & A_j & \longrightarrow & B_kC & (\frac{b_k}{a_j}) & (6) & A & \longrightarrow & B_kC & (\frac{b_k}{a}) \\
(3) & A_j & \longrightarrow & BC_l & (\frac{c_l}{a_j}) & (7) & A & \longrightarrow & BC_l & (\frac{c_l}{a}) \\
(4) & A_j & \longrightarrow & B_kC_l & (\frac{b_k c_l}{a_j}) & (8) & A & \longrightarrow & B_kC_l & (\frac{b_k c_l}{a})
\end{array}
\tag{2.19}
$$

These rules correspond to the eight possible contexts in which the node grouping in example (2.18) can occur in fragments extracted from the corresponding treebank tree; each of the three nodes can be either interior or exterior (i.e. root node or substitution site) to any fragment in which the grouping occurs. The examples in (2.20) illustrate the contexts to which rules (3) – (6) in example (2.19) correspond. Node $A@j$ is an interior (i.e. non-root) node in rules 3 and 4 and an exterior (i.e. root) node in rules 5 and 6 – the parent node of any grouping (the node which appears on the left-hand side of the rule) corresponds to either a root or internal node but not a substitution site. Conversely, the child nodes of each grouping, which appear on the right-hand side of the corresponding rules, can be either internal nodes or substitution sites but never root nodes as shown in example (2.20).

As previously stated, Goodman's PCFG reduction requires the projection of *at most* eight rules for each node in the treebank. The maximum number of rules are projected from each node which is internal to a treebank tree and dominates two non-terminal children; four rules are projected from each node corresponding to the root node of a treebank tree, as this node can never be internal to a fragment, and two rules are projected from nodes dominating a single terminal symbol as terminal symbols are never substitution sites.

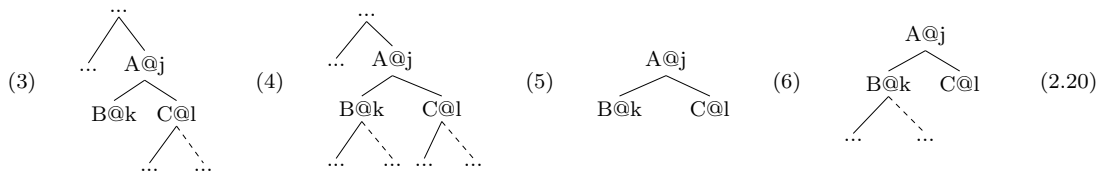
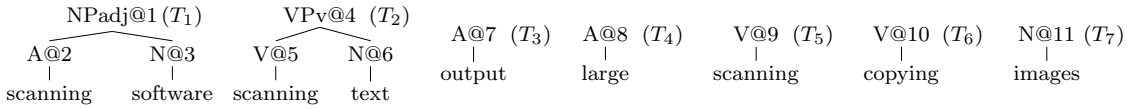


Figure 2.9 illustrates the process of constructing the parse space for an input string using Goodman's method – the treebank with node addresses marked is given in 2.9(A), Goodman's PCFG reduction of the treebank is given in 2.9(B) and the corresponding parse space for the input string *scanning images* is given in 2.9(C). Figure 2.9(D) gives the four parse tree derivations, along with their probabilities, which can be extracted from this parse space.

Goodman states that a PCFG derivation is isomorphic to a DOP derivation if for every substitution of a DOP fragment there is a corresponding sub-derivation in the PCFG. In other words, each PCFG sub-derivation yielding a subtree whose internal nodes are all of the form X_y, whose root node is of the form X and whose frontier nodes are either of the form X or are terminal symbols, corresponds exactly to a DOP fragment when the subscripts are removed. Furthermore, each such PCFG sub-derivation has exactly the same probability as the DOP fragment to which it corresponds. In the example shown in Figure 2.9(D) and (E), each PCFG derivation corresponds to an STSG derivation for the same string where the STSG was extracted from the same treebank (this example corresponds exactly to the one given in Figure 2.3). However, this one-to-one correspondence only occurs where all node groupings (and, therefore, fragments) occur exactly once in the treebank; where node groupings occur more than once, one must sum over several PCFG derivation probabilities for each STSG derivation.

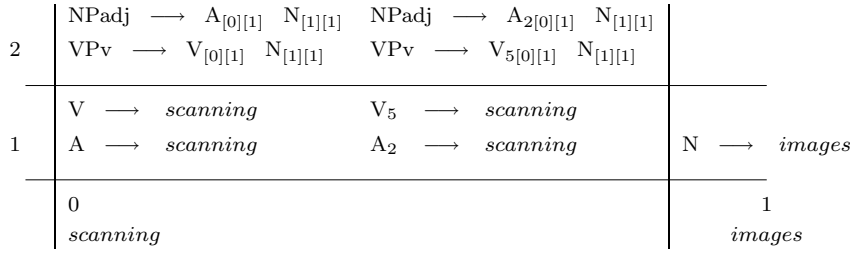
(**A**) **The sample treebank given in Figure 2.3; here, each node is labelled with a unique address** :



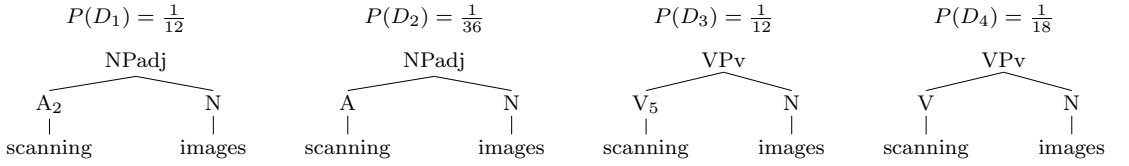
(**B**) **The Goodman PCFG reduction corresponding to the treebank in** (**A**) :

NPadj	$\longrightarrow$	A	N	$\frac{1}{npadj} = \frac{1}{4}$	VPv	$\longrightarrow$	V	N	$\frac{1}{vpv} = \frac{1}{4}$
NPadj	$\longrightarrow$	A_2	N	$\frac{a_2}{npadj} = \frac{1}{4}$	VPv	$\longrightarrow$	V_5	N	$\frac{v_5}{vpv} = \frac{1}{4}$
NPadj	$\longrightarrow$	A	N_3	$\frac{n_3}{npadj} = \frac{1}{4}$	VPv	$\longrightarrow$	V	N_6	$\frac{n_6}{vpv} = \frac{1}{4}$
NPadj	$\longrightarrow$	A_2	N_3	$\frac{a_2 n_3}{npadj} = \frac{1}{4}$	VPv	$\longrightarrow$	V_5	N_6	$\frac{v_5 n_6}{vpv} = \frac{1}{4}$
A	$\longrightarrow$	scanning		$\frac{1}{a} = \frac{1}{3}$	A_2	$\longrightarrow$	scanning		$\frac{1}{a_2} = 1$
A	$\longrightarrow$	output		$\frac{1}{a} = \frac{1}{3}$					
A	$\longrightarrow$	large		$\frac{1}{a} = \frac{1}{3}$					
N	$\longrightarrow$	software		$\frac{1}{n} = \frac{1}{3}$	N_3	$\longrightarrow$	software		$\frac{1}{n_3} = 1$
N	$\longrightarrow$	text		$\frac{1}{n} = \frac{1}{3}$	N_6	$\longrightarrow$	text		$\frac{1}{n_6} = 1$
N	$\longrightarrow$	images		$\frac{1}{n} = \frac{1}{3}$					
V	$\longrightarrow$	copying		$\frac{1}{v} = \frac{1}{3}$					
V	$\longrightarrow$	scanning		$\frac{1}{v} + \frac{1}{v} = \frac{2}{3}$	V_5	$\longrightarrow$	scanning		$\frac{1}{v_5} = 1$

(**C**) **The parse space for the string** *scanning images* **according to the grammar in** (**B**) :



(**D**) **The derivations** (**and their probabilities**) **which can be read from the parse space in** (**C**) :



(**E**) **The STSG derivations corresponding to the PCFG derivations in** (**D**) :

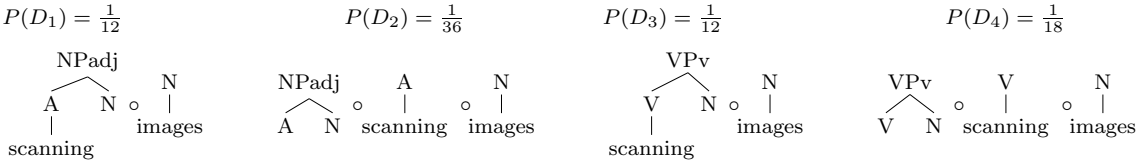
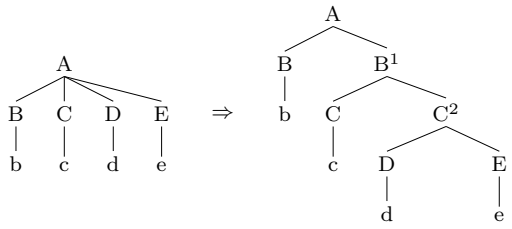


Figure 2.9: Calculation of the parse space for the string *scanning images* using Goodman (1996a, 1998, 2003)'s PCFG-reduction.

Clearly, although it yields the same trees with the same probabilities, the parse space corresponding to the PCFG-reduction does not look the same as the DOP parse space – it contains CFG rules rather than fragments and these rules specify syntactic categories which do not appear in the corresponding DOP grammar. Conversion to the DOP parse space is possible but computationally expensive; as Goodman introduces a novel disambiguation procedure which does not require this conversion (and which will be discussed in section 2.5.6), he does not discuss this issue.

2.4.4 Extended Chomsky-Normal Form

A grammar rule is in CNF if each of its rules has a single non-terminal symbol on the left-hand side and either two non-terminal symbols or a single terminal symbol on the right-hand side. Sima'an (1999) describes a method by which a treebank can be converted to Extended Chomsky-Normal Form (ECNF) such that every tree in the converted treebank is maximally binary branching.



(2.21)

This method requires the insertion of new, unique node categories into treebank trees at each node whose branching factor is greater than 2, as illustrated in example (2.21). This process does not affect the DOP probability model as no newly-inserted node is allowed to be either the root node or substitution site of any fragment. Furthermore, the original treebank trees and/or DOP fragments can be easily recovered by simply reversing the process and removing the inserted nodes.

2.5 Tree-DOP disambiguation strategies

The first task when parsing an input string over a DOP grammar is to compute a compact representation of all possible parses which can be generated for that string according to the

grammar; in the previous section (section 2.4), we outlined three methods for achieving this goal. The next task, that of disambiguation, is to rank these parses according to the DOP probability model. However, ranking the parses for a given input string over a DOP grammar has been shown to be an NP-hard problem (Sima'an, 1995b, 1999, 2003) as computing the probability of a parse involves summing over the probabilities of all derivations yielding that parse. As its exact solution cannot be found in an efficient way, we must either find a way of approximating the search for the most probable parse such that we do not perform an exhaustive search of the parse space, or we must choose a different probability to maximise. Several approaches have been proposed; approximation of the most probable parse by random sampling is discussed in section 2.5.1 while alternatives to the maximisation of parse probability are presented in sections 2.5.2 – 2.5.6.[13]

2.5.1 Most Probable Parse

Monte Carlo sampling can be used to estimate the Most Probable Parse (MPP) where the DOP-ranked list of parses is approximated by ranking according to how often each parse occurs in a reduced random sample of the possible derivations. This approach to disambiguation for DOP was introduced by Bod (1992) and further expanded on and refined by Chappelier and Rajman (2003).

The basic premise behind the application of Monte Carlo estimation to MPP extraction is as follows (Chappelier and Rajman, 2003):

> given a set of random derivations with a known sampling distribution, the proportion of derivations in the set corresponding to parse P will converge to the sum of the sampling probabilities of all derivations of P.

However, this convergence property can only be used to estimate P_{DOP} of parse tree P if the following condition is fulfilled:

[13] de Pauw (2003) presents an approximation of the DOP model through Memory-Based Language Processing (MBLP) (Daelemans, 1999), in which the memory-based aspect of the model is exploited. Under this model, a parse forest for each input string is generated using the grammar underlying the training treebank. The best parse is then determined by directly comparing the constituents of each parse with the constituents occurring in the treebank trees, and calculating similarity such that the number of constituents required to construct the tree is minimised and the size of those constituents is maximised. Although interesting, further exploration of this approach is beyond the scope of this thesis.

$P_{DOP}(P)$ can be estimated from the set of sampled derivations, i.e. there exists a function which can be computed over the sample set which converges to its P_{DOP} when the number of samples grows to infinity.

In other words, we can only use sampling frequencies to rank parses if we can establish the relationship between the sampling frequency of a parse and its DOP probability.

Sampling Algorithm

The sampling methodology itself is very simple: in order to sample a derivation, we select and compose fragments at random from the parse space in a top-down left-to-right fashion until no open substitution sites remain. This can be done efficiently when the open substitution sites of each fragment are annotated with pointers to the chart position from which fragments which compose with it should be selected.

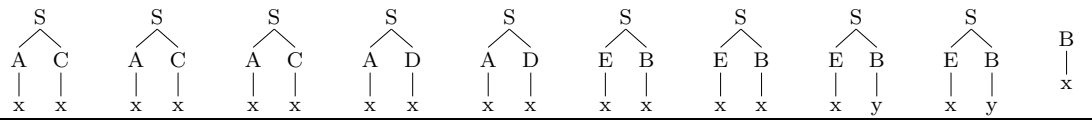
However, we must select fragments at random such that, if the sampling probability of fragment f_x is n times that of f_y, then f_x is n times more likely to be chosen during random selection than f_y. The main issue, therefore, is to correctly define the sampling probability of each fragment at each chart position $SP(f_{ij})$ such that the distribution of parses in the sample set converges to the true P_{DOP}.

The sampling probability used when selecting fragments can be defined in advance very easily. If we choose to do this, however, then we cannot be certain that the distribution of the sample set will converge to give the DOP probability for each parse. The correct values must instead be obtained by *rescoring* the relative frequencies of the parses in the sample set when sampling is complete. Thus the empirical score ES of parse tree P is defined by equation (2.22), where n is the size of the sample set and W_i is a rescoring factor.

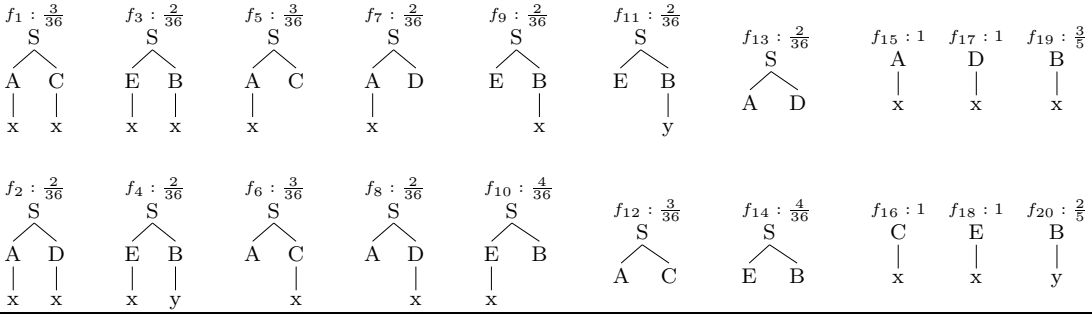
$$ES(P) = \sum_{D_i \ yields \ P} \frac{n_i}{n} W_i \tag{2.22}$$

Alternatively, the sampling probability of each fragment can be computed such that the sampling probability of each parse in the sample set corresponds exactly to its conditional DOP probability. Although less simple to implement, computing these probabilities allows faster convergence and precise control over the size of the sample set.

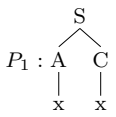
(**A**) **A sample treebank** :



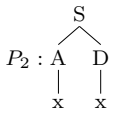
(**B**) **The DOP fragment base generated from the treebank in** (**A**) :



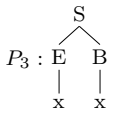
(**D**) **Parse probabilities** (**conditioned on the input string**) **computed from** (**B**) **for string** xx:



$$P_1 = P(f_1) + P(f_5 \circ f_{16}) + P(f_6 \circ f_{15}) + P(f_{12} \circ f_{15} \circ f_{16}) =$$
$$\frac{3}{36} + \frac{3}{36}.1 + \frac{3}{36}.1 + \frac{3}{36}.1.1 = \frac{12}{36} = \frac{60}{180}.\frac{180}{144} = \frac{60}{144}$$



$$P_2 = P(f_2) + P(f_7 \circ f_{17}) + P(f_8 \circ f_{15}) + P(f_{13} \circ f_{15} \circ f_{17}) =$$
$$\frac{2}{36} + \frac{2}{36}.1 + \frac{2}{36}.1 + \frac{2}{36}.1.1 = \frac{8}{36} = \frac{40}{180}.\frac{180}{144} = \frac{40}{144}$$



$$P_3 = P(f_3) + P(f_9 \circ f_{18}) + P(f_{10} \circ f_{19}) + P(f_{14} \circ f_{18} \circ f_{19}) =$$
$$\frac{2}{36} + \frac{2}{36}.1 + \frac{4}{36}.\frac{3}{5} + \frac{4}{36}.1.\frac{3}{5} = \frac{44}{180} = \frac{44}{180}.\frac{180}{144} = \frac{44}{144}$$

Figure 2.10: (C) gives the DOP distribution of the three parses for string xx generated by the grammar in (B).

Again, the goal when sampling DOP derivations is to sample such that the distribution of parse trees in the sampled set corresponds to their DOP probability distribution. In Figure 2.10, we give an example treebank, the DOP fragments and their relative frequencies yielded by that treebank, and the parse tree probabilities for the string *xx* according to the DOP model. We see from the example that there are three possible parses – P_1, P_2 and P_3 – for string *xx* according to the grammar, each of which can be derived in four different ways. According to the DOP probability model, parse P_1 is most likely with probability $\frac{60}{144}$, followed by P_3 with probability $\frac{44}{144}$ and finally P_2 with probability $\frac{40}{144}$. (These probabilities are conditioned on the input string, i.e. we divide each parse probability by the total probability mass assigned to all parses of that string.) We require that the parses in the sample set for this parse space also conform to this distribution.

We present the rescored and exact approaches to sampling in the remainder of this section. In order to illustrate how the choice of sampling probability calculation affects the sampling distribution, we give the sampling parse spaces for the same string and DOP fragment set as in Figure 2.10 for each method, and show the corresponding sampling distributions.

Rescored Sampling: Naïve A first, naïve solution is to simply assume that each fragment in competition set CS has equal probability of being selected at random. Without rescoring, the sampling probability for each parse is calculated according to equation (2.23).

$$SP(P) = \sum_{D \; yields \; P} \prod_{f_{ij} \; \epsilon \; D} \frac{1}{|CS_{ij}|} \tag{2.23}$$

As is clear from the example in Figure 2.11, the set of sampled parses generated according to these probabilities is not distributed according to the DOP probability model. As can be seen from the parse sampling probabilities (given in brackets), the sampling distribution indicates that all three parses for the input string are equally likely. This can be corrected by applying a rescoring factor W calculated according to equation (2.24).

$$W_D = \frac{P_{DOP}(D)}{\prod_{f_{ij} \; \epsilon \; D} \frac{1}{|CS_{ij}|}} \tag{2.24}$$

2	$SP(f_1)=\frac{1}{12}$	$SP(f_6)=\frac{1}{12}$	$SP(f_{10})=\frac{1}{12}$			
	$SP(f_2)=\frac{1}{12}$	$SP(f_7)=\frac{1}{12}$	$SP(f_{12})=\frac{1}{12}$			
	$SP(f_3)=\frac{1}{12}$	$SP(f_8)=\frac{1}{12}$	$SP(f_{13})=\frac{1}{12}$			
	$SP(f_5)=\frac{1}{12}$	$SP(f_9)=\frac{1}{12}$	$SP(f_{14})=\frac{1}{12}$			
1	$SP(f_{15})=1$	$SP(f_{17})=1$	$SP(f_{19})=1$	$SP(f_{15})=1$	$SP(f_{17})=1$	$SP(f_{19})=1$
	$SP(f_{16})=1$	$SP(f_{18})=1$		$SP(f_{16})=1$	$SP(f_{18})=1$	
	0			1		
	x			x		

P_1 : S(A, C)	$=$	$SP(f_1)$	$+$	$SP(f_5).SP(f_{16})$	$+$	$SP(f_6).SP(f_{15})$	$+$	$SP(f_{12}).SP(f_{15}).SP(f_{16})$	$=$
		$\frac{1}{12}$	$+$	$\frac{1}{12}.1$	$+$	$\frac{1}{12}.1$	$+$	$\frac{1}{12}.1.1$	$(=\frac{48}{144})$
	$W:$	$\uparrow.\frac{3}{36}.12$		$\uparrow.\frac{3}{36}.12.1$		$\uparrow.\frac{3}{36}.12.1$		$\uparrow.\frac{3}{36}.12.1$	$=\frac{60}{180}.\frac{180}{144}=\frac{60}{144}$
P_2 : S(A, D)	$=$	$SP(f_2)$	$+$	$SP(f_7).SP(f_{17})$	$+$	$SP(f_8).SP(f_{15})$	$+$	$SP(f_{13}).SP(f_{15}).SP(f_{17})$	$=$
		$\frac{1}{12}$	$+$	$\frac{1}{12}.1$	$+$	$\frac{1}{12}.1$	$+$	$\frac{1}{12}.1.1$	$(=\frac{48}{144})$
	$W:$	$\uparrow.\frac{2}{36}.12$		$\uparrow.\frac{2}{36}.12.1$		$\uparrow.\frac{2}{36}.12.1$		$\uparrow.\frac{2}{36}.12.1$	$=\frac{40}{180}.\frac{180}{144}=\frac{40}{144}$
P_3 : S(E, B)	$=$	$SP(f_3)$	$+$	$SP(f_9).SP(f_{18})$	$+$	$SP(f_{10}).SP(f_{19})$	$+$	$SP(f_{14}).SP(f_{18}).SP(f_{19})$	$=$
		$\frac{1}{12}$	$+$	$\frac{1}{12}.1$	$+$	$\frac{1}{12}.1$	$+$	$\frac{1}{12}.1.1$	$(=\frac{48}{144})$
	$W:$	$\uparrow.\frac{2}{36}.12$		$\uparrow.\frac{4}{36}.12.\frac{3}{5}$		$\uparrow.\frac{2}{36}.12.1$		$\uparrow.\frac{4}{36}.12.\frac{3}{5}$	$=\frac{44}{180}.\frac{180}{144}=\frac{44}{144}$

Figure 2.11: Sampling distribution induced by naïve rescored sampling over the example given in Figure 2.10.

Thus, when we apply rescoring factor W to each derivation sampling probability given in Figure 2.11 – application of W is indicated using up arrows ($\uparrow$) – and recalculate the parse probabilities according to equation (2.22), we arrive at the correct DOP distribution for all parses of the input string.

Rescored Sampling: Hoogweg Hoogweg (2000)'s sampling technique is exactly that of the naïve approach described in the previous section but with a different sampling probability distribution. He takes the sampling probability of a fragment to be its DOP probability[14] over the sum of the DOP probabilities of the fragments it is competing with

[14]Throughout this discussion, the parameter estimation method used is assumed to be the relative frequency estimator.

2	$SP(f_1) = \frac{3}{36}.\frac{36}{32} = \frac{3}{32}$ $SP(f_8) = \frac{2}{36}.\frac{36}{32} = \frac{2}{32}$ $SP(f_2) = \frac{2}{36}.\frac{36}{32} = \frac{2}{32}$ $SP(f_9) = \frac{2}{36}.\frac{36}{32} = \frac{2}{32}$ $SP(f_3) = \frac{2}{36}.\frac{36}{32} = \frac{2}{32}$ $SP(f_{10}) = \frac{4}{36}.\frac{36}{32} = \frac{4}{32}$ $SP(f_5) = \frac{3}{36}.\frac{36}{32} = \frac{3}{32}$ $SP(f_{12}) = \frac{3}{36}.\frac{36}{32} = \frac{3}{32}$ $SP(f_6) = \frac{3}{36}.\frac{36}{32} = \frac{3}{32}$ $SP(f_{13}) = \frac{2}{36}.\frac{36}{32} = \frac{2}{32}$ $SP(f_7) = \frac{2}{36}.\frac{36}{32} = \frac{2}{32}$ $SP(f_{14}) = \frac{4}{36}.\frac{36}{32} = \frac{4}{32}$		
	$SP(f_{15}) = \frac{1}{1} = 1$ $SP(f_{18}) = \frac{1}{1} = 1$ $SP(f_{16}) = \frac{1}{1} = 1$ $SP(f_{19}) = \frac{3}{5}.\frac{5}{3} = 1$ $SP(f_{17}) = \frac{1}{1} = 1$	$SP(f_{15}) = \frac{1}{1} = 1$ $SP(f_{18}) = \frac{1}{1} = 1$ $SP(f_{16}) = \frac{1}{1} = 1$ $SP(f_{19}) = \frac{3}{5}.\frac{5}{3} = 1$ $SP(f_{17}) = \frac{1}{1} = 1$	
	0 x	1 x	

P_1 : (S (A x) (C x))	=	$SP(f_1)$	+	$SP(f_5).SP(f_{16})$	+	$SP(f_6).SP(f_{15})$	+	$SP(f_{12}).SP(f_{15}).SP(f_{16})$	=
		$\frac{3}{32}$	+	$\frac{3}{32}.1$	+	$\frac{3}{32}.1$	+	$\frac{3}{32}.1.1$	$(= \frac{54}{144})$
	W :	$\uparrow.\frac{32}{36}$		$\uparrow.\frac{32}{36}.1$		$\uparrow.\frac{32}{36}.1$		$\uparrow.\frac{32}{36}.1.1$	$= \frac{60}{180}.\frac{180}{144} = \frac{60}{144}$
P_2 : (S (A x) (D x))	=	$SP(f_2)$	+	$SP(f_7).SP(f_{17})$	+	$SP(f_8).SP(f_{15})$	+	$SP(f_{13}).SP(f_{15}).SP(f_{17})$	=
		$\frac{2}{32}$	+	$\frac{2}{32}.1$	+	$\frac{2}{32}.1$	+	$\frac{2}{32}.1.1$	$(= \frac{36}{144})$
	W :	$\uparrow.\frac{32}{36}$		$\uparrow.\frac{32}{36}.1$		$\uparrow.\frac{32}{36}.1$		$\uparrow.\frac{32}{36}.1.1$	$= \frac{40}{180}.\frac{180}{144} = \frac{40}{144}$
P_3 : (S (E x) (B x))	=	$SP(f_3)$	+	$SP(f_9).SP(f_{18})$	+	$SP(f_{10}).SP(f_{19})$	+	$SP(f_{14}).SP(f_{18}).SP(f_{19})$	=
		$\frac{2}{32}$	+	$\frac{2}{32}.1$	+	$\frac{4}{32}.1$	+	$\frac{4}{32}.1.1$	$(= \frac{54}{144})$
	W :	$\uparrow.\frac{32}{36}$		$\uparrow.\frac{32}{36}.\frac{3}{5}$		$\uparrow.\frac{32}{36}.1$		$\uparrow.\frac{32}{36}.1.\frac{3}{5}$	$= \frac{44}{180}.\frac{180}{144} = \frac{44}{144}$

Figure 2.12: Sampling distribution induced by Hoogweg rescored sampling over the example given in Figure 2.10.

for selection, as given in equation (2.25).

$$SP(P) = \sum_{D \; yields \; P} \prod_{f_{ij} \; \epsilon \; D} \frac{P(f_{ij})}{\sum_{f' \; \epsilon \; CS_{ij}} P(f')} \tag{2.25}$$

Calculation of these sampling probabilities is illustrated in Figure 2.12; again we see that the the sampling distribution induced (given in brackets for each parse) does not correspond to the DOP probability distribution of the parse trees assigned to the input string. This time, rescoring factor W is is calculated according to equation (2.26) and the

probability of each parse recalculated according to equation (2.22).

$$W_D = \prod_{f_{ij} \,\epsilon\, D} \sum_{f' \,\epsilon\, CS_{ij}} P(f') \tag{2.26}$$

Again, application of the required rescoring factors to the sampled derivations in Figure 2.12 yields the desired distribution.

Bod (1998) computes sampling probabilities using the same formula as Hoogweg (equation (2.25)) but uses rescoring factor $W_D = 1$. As pointed out by Chappelier and Rajman (2003), and as is clearly illustrated by the example we give in Figure 2.12, this does not lead to the correct probabilities and, therefore, imposes a ranking on the parse trees generated by the DOP grammar which does not correspond to the DOP probability model.

Exact Sampling The purpose of exact sampling is to ensure that the sampling probability of each parse tree is directly equal to the conditional DOP probability of that parse given the input string. As stated by Chappelier and Rajman (2003), this technique then guarantees that the best parse tree also has the best sampling probability and that the most frequent tree in a sampled set has a high probability of being the MPP. This method also enables statistical control over the size of the sampled set.

The formula which must be used to calculate the sampling probability of each fragment so that the final sampling probability of a parse tree corresponds to its DOP probability (conditioned on the input string) is given in (2.27):

$$SP(f_{ij}) = \frac{P(f_{ij}) \prod_{SS_{kl} \,\epsilon\, SSS(f_{ij})} TSP(CS_{kl}(SS))}{TSP(CS_{ij}(root(f_{ij})))} \tag{2.27}$$

Here,

$SSS(f_{ij}) = \{SS_{kl} : f_{ij}$ *has an open substitution site of category* SS
to be filled from chart position $[k][l]\}$,

$CS_{kl}(SS) = \{f : r(f) = SS \wedge f \epsilon [k][l]\}$, and

$TSP(CS_{kl}(SS)) = \sum_{f' \epsilon CS_{kl}(SS)} SP(f')$.

Essentially, formula (2.27) states that the sampling probability of fragment f_{ij} is equal to its DOP probability multiplied by the total sampling probability mass available at each

2	$SP(f_1) = \frac{3}{36}$	$= \frac{3}{36}.\frac{180}{144}$	$= \frac{15}{144}$	
	$SP(f_2) = \frac{2}{36}$	$= \frac{2}{36}.\frac{180}{144}$	$= \frac{10}{144}$	
	$SP(f_3) = \frac{2}{36}$	$= \frac{2}{36}.\frac{180}{144}$	$= \frac{10}{144}$	
	$SP(f_5) = \frac{3}{36}.1$	$= \frac{3}{36}.\frac{180}{144}$	$= \frac{15}{144}$	
	$SP(f_6) = \frac{3}{36}.1$	$= \frac{3}{36}.\frac{180}{144}$	$= \frac{15}{144}$	
	$SP(f_7) = \frac{2}{36}.1$	$= \frac{2}{36}.\frac{180}{144}$	$= \frac{10}{144}$	
	$SP(f_8) = \frac{2}{36}.1$	$= \frac{2}{36}.\frac{180}{144}$	$= \frac{10}{144}$	
	$SP(f_9) = \frac{2}{36}.1$	$= \frac{2}{36}.\frac{180}{144}$	$= \frac{10}{144}$	
	$SP(f_{10}) = \frac{4}{36}.\frac{3}{5}$	$= \frac{12}{180}.\frac{180}{144}$	$= \frac{12}{144}$	
	$SP(f_{12}) = \frac{3}{36}.1.1$	$= \frac{3}{36}.\frac{180}{144}$	$= \frac{15}{144}$	
	$SP(f_{13}) = \frac{2}{36}.1.1$	$= \frac{2}{36}.\frac{180}{144}$	$= \frac{10}{144}$	
	$SP(f_{14}) = \frac{4}{36}.1.\frac{3}{5}$	$= \frac{12}{180}.\frac{180}{144}$	$= \frac{12}{144}$	

1	$SP(f_{15}) = \frac{1}{1} = 1$	$SP(f_{15}) = \frac{1}{1} = 1$
	$SP(f_{16}) = \frac{1}{1} = 1$	$SP(f_{16}) = \frac{1}{1} = 1$
	$SP(f_{17}) = \frac{1}{1} = 1$	$SP(f_{17}) = \frac{1}{1} = 1$
	$SP(f_{18}) = \frac{1}{1} = 1$	$SP(f_{18}) = \frac{1}{1} = 1$
	$SP(f_{19}) = \frac{3}{5}.\frac{5}{3} = 1$	$SP(f_{19}) = \frac{3}{5}.\frac{5}{3} = 1$

	0	1
	x	x

$$P_1 : \begin{array}{c} \text{S} \\ \text{A}\quad\text{C} \\ \text{x}\quad\text{x} \end{array} = SP(f_1) + SP(f_5).SP(f_{16}) + SP(f_6).SP(f_{15}) + SP(f_{12}).SP(f_{15}).SP(f_{16}) =$$
$$\frac{15}{144} + \frac{15}{144}.1 + \frac{15}{144}.1 + \frac{15}{144}.1.1 = \frac{60}{144}$$

$$P_2 : \begin{array}{c} \text{S} \\ \text{A}\quad\text{D} \\ \text{x}\quad\text{x} \end{array} = SP(f_2) + SP(f_7).SP(f_{17}) + SP(f_8).SP(f_{15}) + SP(f_{13}).SP(f_{15}).SP(f_{17}) =$$
$$\frac{10}{144} + \frac{10}{144}.1 + \frac{10}{144}.1 + \frac{10}{144}.1.1 = \frac{40}{144}$$

$$P_3 : \begin{array}{c} \text{S} \\ \text{E}\quad\text{B} \\ \text{x}\quad\text{x} \end{array} = SP(f_3) + SP(f_9).SP(f_{18}) + SP(f_{10}).SP(f_{19}) + SP(f_{14}).SP(f_{18}).SP(f_{19}) =$$
$$\frac{10}{144} + \frac{10}{144}.1 + \frac{12}{144}.1 + \frac{12}{144}.1.1 = \frac{44}{144}$$

Figure 2.13: Sampling distribution induced by exact sampling over the example given in Figure 2.10.

of its substitution sites and divided by the total sampling probability mass available at position $[i][j]$. Chappelier and Rajman prove by induction that using this method, the sampling probability of a derivation $PS(D)$ is equal to its DOP probability conditioned on the input string $\frac{P_{DOP}(D)}{P_{DOP}(w_1^m)}$. This sampling methodology is illustrated in Figure 2.13, where we see that the distribution of parses in the sample set corresponds exactly to the DOP probability distribution of those parse trees conditioned on the input string *without* application of rescoring factors.

Controlling sample size with Exact Sampling

In order to statistically control the size of the sample set, we must determine the minimum number of samples needed to be certain that the most frequent parse in the sample set corresponds to the most probable parse according to the DOP model.

Chappelier and Rajman (2003) restate the problem as follows. Consider input sentence S to be a random variable which has two or more possible values. These values (or modalities) correspond to the two or more parses which have been assigned to S during parsing. We wish to determine the most probable modality P which we can assign to variable S on the basis of a sample of its realisations. Expressed thus, the problem of controlling sample size corresponds to a classical problem of statistical ordering to which we can directly apply a solution from the statistics literature. Here we outline one such solution: Bechhofer-Kiefer-Sobel (BKS) sampling (Chappelier and Rajman, 2003).

BKS is a sequential sampling method, meaning that we continue to sample derivations at random until we fulfil a stopping condition which is predefined but recalculated each time a sample is taken. BKS relies on the following: for any input sentence S with k parses ($< p_{[1]}...p_{[k]} >$) such that $p_{[1]} \geq \theta p_{[2]}$ with $\theta > 1$, the probability that the most frequent parse in the sample is also the most probable one is always greater than $\frac{1}{1+Z}$ where $Z = \sum_{i=2}^{k} (\frac{1}{\theta})^{(n_{[1]} - n_{[i]})}$ and where $n_{[i]}$ is the number of occurrences of the parse in ith position on the ordered list (decreasing order) of parses seen. The BKS method is then:

- choose values for $\theta = \frac{p_{[1]}}{p_{[i]}}$ and the error probability P_{err},
- sample – updating the ordered list of parses and their frequencies and Z – until $\frac{1}{1+Z} \geq P_{err}$,

- output the most frequent parse in the sample as the most probable one.

The decision to stop sampling is thus based on three factors:

- how closely matched, in terms of frequency of occurrence, the parses in the sample set are,
- how many of the possible parses for the given input string are present in the set of sampled parses, and
- how certain we wish to be that the most frequent parse in the sample set is in fact the most probable parse according to the DOP model.

Controlling sample size according to Bod (1998) Bod (1998) controls the size of the sample set by calculating in advance the number of samples to be taken. He firstly defines the probability of error (i.e. the probability that the most frequent sample is not, in fact, the most probable) as $\sum_{i \neq 0}(1 - (\sqrt{p_0} - \sqrt{p_i})^2)^N$. He then states that if we try to estimate each parse probability by its frequency in the sample set, the variance in the probability estimate is $\frac{p_i(1-p_i)}{N}$. As p_i always lies between 0 and 1, the maximum variance is $\frac{1}{4N}$. The standard error σ, which is calculated as the square root of the variance, thus has a maximum value of $\frac{1}{2\sqrt{N}}$. If we define in advance an upper bound for σ (again, the probability that our ranking is incorrect), then we can calculate a lower bound for N. N is the smallest integer larger than $\frac{1}{4\sigma_2}$. For example, if $\sigma \leq 0.05$ then $N \geq 100$ or if $\sigma \leq 0.01$ then $N \geq 2500$.

Chappelier and Rajman (2003) state that Bod's method of controlling the sample size is wrong because the sampling probabilities themselves are wrong yet Bod estimates the error probability $\sum_{i \neq 0}(1 - (\sqrt{p_0} - \sqrt{p_i})^2)^N$ by $\sum_{i \neq 0}(1 - (\sqrt{f_0} - \sqrt{f_i})^2)^N$. Furthermore, they say that the sequential nature of his method does not permit advance computation of the minimum sample size.

2.5.2 Most Probable Derivation

In PCFG parsing, each unique parse for a given input string is generated by exactly one derivation, which means that calculating the Most Probable Derivation (MPD) is

equivalent to calculating the MPP. There exists an efficient algorithm, called Viterbi optimisation, to compute the MPD for a PCFG. Viterbi optimisation involves pruning sub-derivations with low probabilities from the PCFG parse space in a bottom-up manner. Two different sub-derivations which have the same root node and span the same portion of the input string are used in building derivations of the entire input string in exactly the same way. This means that parses containing the more probable of these sub-derivations will always be more probable than those parses containing the less probable sub-derivation. Consequently, the less probable sub-derivation will never be used to build the most probable derivation/parse and can be removed from the parse space.

Viterbi optimisation can also be used to compute the MPD for DOP grammars. However, DOP models differ from PCFG models in that many different DOP derivations can yield exactly the same parse tree, and the probability of a parse is calculated by summing over the probabilities of all derivations which yield it. It is not possible to use Viterbi optimisation when computing the MPP as sub-derivations which are less likely are pruned from the chart regardless of the fact that their probabilities may contribute to that of the MPP (Bod, 1995b).

As it is not feasible to compute exactly the MPP for DOP and it may be less computationally expensive to compute exactly the MPD than to approximate the MPP using random sampling, it is important to look at parse accuracy when maximising derivation probability as opposed to parse probability. Bod (2003c) presents empirical evidence that maximising derivation probability rather than parse probability results in a 16% decrease in parse accuracy[15] and concludes that, at least for the ATIS treebank, it is more useful to search for the MPP. This is unsurprising as computation of the MPP allows for overlapping fragments: every fragment which can be used to parse the input string provides probabilistic evidence in favour of those parses it helps to derive.

2.5.3 Most Probable Parse amongst the n Most Probable Derivations

Bod (2000e) presents experiments which were carried out using Monte Carlo disambiguation on the ATIS corpus. He then repeated these experiments using an alternative dis-

[15] In these experiments, parse accuracy is measured as the proportion of test set sentences whose output parses exactly match their corresponding test set parses in the treebank.

ambiguation strategy, where the MPP is selected by computing the 1000 most probable derivations using the Viterbi n-best optimisation and then explicitly summing over these derivation probabilities to compute the most probable parse. His results show that the two methods achieved similar scores; the differences were not statistically significant. Bod also used this disambiguation method when parsing the Penn-II Treebank and achieved an 11% relative error reduction over previous models (Bod, 2003a).

2.5.4 Simplicity: the shortest derivation

PCFG parse probability is computed by multiplying the probabilities of each rule used to derive a parse. Parses derived using fewer rules generally have higher probabilities than parses of the same sentence which use greater numbers of rules. This results in an undesirable bias towards smaller parse trees. However, Bod (2000e) observes that because DOP grammars use fragments of varying sizes, the tendency to favour shorter derivations does not necessarily result in a bias towards smaller parse trees. Rather, the shortest DOP derivation of a parse tree is built using the *largest* relevant fragments in the fragment base.

Bod (2000e) investigates possible benefits of this bias in favour of shorter derivations by defining a DOP model in which the best parse tree is the one which is built using the smallest number of fragments F_{min}, i.e. by the shortest derivation. Derivation lengths are computed by assigning each fragment equal probability, meaning that the shortest derivation can be computed as the most probable one using Viterbi.[16]

Although Bod describes this approach as a non-probabilistic DOP model, in the case where more than one parse tree can be derived using F_{min} fragments, fragment probabilities determine which of these parses is selected as the best parse. In order to back off to frequency-ordering of the fragments, each is assigned a rank according to its frequency in the fragment base. The most frequent fragment of each root label is assigned rank 1, the second most frequent rank 2 and so on. The ranks of those derivations using F_{min} fragments are computed as the sum of the ranks of the fragments used in building each derivation; the highest-ranked derivation (i.e. the one with the smallest sum) is deemed the shortest derivation and, therefore, yields the best parse tree.

[16] If each fragment has probability p then the probability of a derivation which uses n fragments is p^n; since $0 < p < 1$, the smallest n has the largest probability.

Evaluation of this model was carried out on the ATIS, OVIS and WSJ treebanks; comparison was against a model which disambiguated using the Viterbi n-best technique described in section 2.5.3. Experiments on the ATIS treebank showed that the shortest derivation model performed significantly worse than the Viterbi n-best model at lower fragment depths; by depth 4, scores are roughly the same and at depth 6 the shortest derivation model outperforms Viterbi n-best by 1.5%. Experiments on the OVIS treebank again showed that the shortest derivation model performed poorly at lower fragment depths. However, probabilistic DOP suffers from serious data-sparseness on the OVIS treebank and accuracy decreases as fragment depth exceeds 4. The accuracy of the shortest derivation model, however, continues to rise (at depth 6, it outperforms the Viterbi n-best model by 3.4%) which, according to Bod, would appear to indicate that this model is less sensitive to low frequency counts of larger fragments. Experiments on the WSJ show that, while the results are comparable, the Viterbi n-best model consistently outperforms the shortest derivation model.

2.5.5 Simplicity and Likelihood Combined

Bod (2003a) outlines two models which combine the shortest derivation model described in section 2.5.4,[17] referred to as Simplicity-DOP, with the Viterbi n-best model described in section 2.5.3 which he refers to as Likelihood-DOP. Although Simplicity-DOP does not perform as well as Likelihood-DOP (the best simplicity f-score is 2.3% lower than the best likelihood f-score on the WSJ corpus (Bod, 2000e)), it achieves very good results for such a simple model. Furthermore, according to Bod (2003a), Simplicity-DOP selects parse trees which are quite different from those selected by Likelihood-DOP, suggesting that combined models may improve performance.

The idea behind the first model, termed *Simplicity-Likelihood-DOP* or SL-DOP, is to select the simplest parse tree from among the n likeliest trees. The second model, which selects the likeliest tree from among the n simplest trees, is referred to as *Likelihood-Simplicity-DOP* or LS-DOP. SL-DOP is equal to Likelihood-DOP when $n = 1$ as there is only one most likely parse to choose from; as n gets large the model converges to Simplicity-

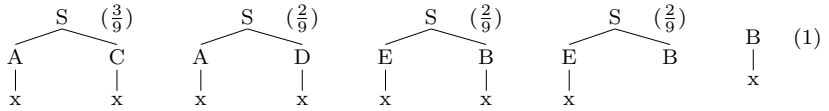
[17] One adjustment was made: the rank of each fragment was averaged by the rank of all its sub-fragments.

DOP as the simplest parse is chosen from among most or all of the likely parses. The opposite clearly holds for LS-DOP.

These models were compared by performing experiments on the WSJ treebank[18] where the value for n varied such that $1 \leq n \leq 1,000$. Results indicate that SL-DOP achieves greatest accuracy at $12 \leq n \leq 14$, thereafter decreasing and converging to Simplicity-DOP, whereas the accuracy of LS-DOP continues to increase as n increases and converges to Likelihood-DOP. The best results are obtained by SL-DOP at $12 \leq n \leq 14$; these parsing results are the highest published for the WSJ treebank.

2.5.6 Maximum Constituents Parse

Goodman (1996b) observes that although many different metrics exist for evaluating parsing results (e.g. exact match, labelled precision and recall, unlabelled precision and recall, crossing brackets rate), most parsing algorithms seek to optimise just one metric: exact match. He contends that better parsing results can be achieved by developing disambiguation algorithms appropriate to the evaluation metric being used. The majority of parsing metrics can be said to calculate roughly the number of correct constituents[19] in each output parse (the obvious exception being exact match). According to Goodman (1996a, 1998, 2003), if a parser is to be evaluated on such a measure then the parser should output the parse most likely to have the largest number of correct constituents, i.e. the Maximum Constituents Parse (MCP). Goodman illustrates this maximisation with the following example. Assume the following DOP grammar in (2.28):

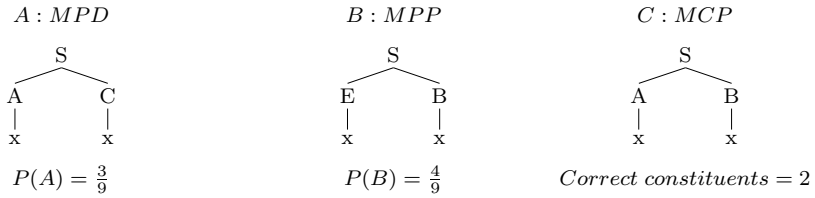


(2.28)

The three three best parses in this grammar according to three different criteria are those in (2.29):

[18] Similar experiments are presented in (Bod, 2002) but maximum fragment depth was limited to 14; in the experiments currently under discussion (Bod, 2003a), all fragments were used.

[19] A constituent is a triple (i, X, j) where X is a non-terminal which dominates words $w_i ... w_j$ of an input string which spans $w_1 ... w_n$. A *labelled* constituent is correct if the treebank parse also contains the constituent (i, X, j) and an *unlabelled* constituent is correct if the treebank parse contains any constituent (i, Y, j) which spans $w_i ... w_j$ regardless of its non-terminal symbol.



(2.29)

When calculating the MCP, we need to know how many constituents of each parse we expect to be correct; the parse with the greatest number of expected correct constituents is the MCP. For parse (C) above, the probability that the constituent $(0, S, 1)$ is correct is 1 because the only non-terminal which can span the full input string (i.e. two words) is S. The probability that the chosen constituent $(0, A, 0)$ is correct is $\frac{5}{9}$; the only other non-terminal which can span this section of the input string (i.e. the first word) is E with probability $\frac{4}{9}$. Finally, the probability that the chosen constituent $(1, B, 1)$ is correct is $\frac{4}{9}$; non-terminals C and D can span the same word (i.e. the second word) with probabilities $\frac{3}{9}$ and $\frac{2}{9}$ respectively. It follows that the chosen tree has two correct constituents; all other trees have fewer than two correct constituents so this parse is selected as the MCP. Goodman's algorithm uses inside-outside values to calculate the probability that the constituent is in the correct parse and then uses dynamic programming to put the most likely constituents together to form an output parse tree.

It is clear from the example given above that this model is not a special case of the DOP model as the MCP can never be produced by combining DOP fragments. Nonetheless, Goodman (2003) presents experimental results which demonstrate that outputting the MCP yields similar performance to outputting the DOP MPP using Monte Carlo estimation.

2.6 Estimating Tree-DOP fragment probabilities

The method of estimating fragment probabilities described in section 2.2.4 – whereby the probability of a fragment is taken to be its relative frequency amongst all fragments in the fragment set which have the same root node as it – has been shown to be an unsatisfactory parameter estimation method for DOP (Bonnema et al., 2000; Johnson, 2002). In this

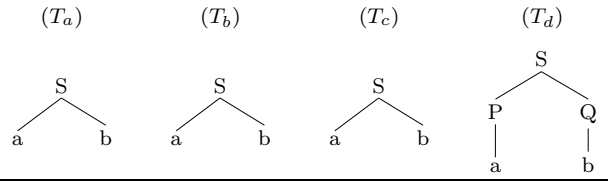
section, we describe why this is the case using a simple example, and outline three solutions which seek to address this problem.[20]

2.6.1 The relative frequency estimator: DOP$_{rf}$

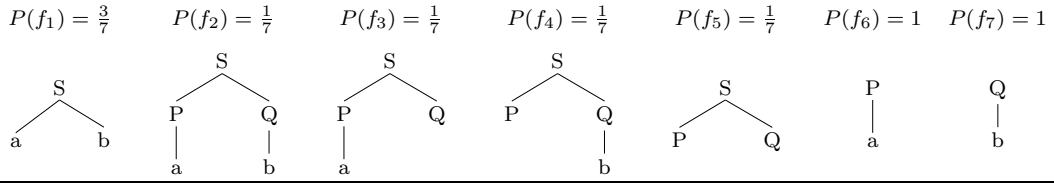
The relative frequency estimator for the DOP model (DOP$_{rf}$) induces a bias in favour of larger parse trees; this issue was first identified in (Bonnema et al., 2000) and further examined in (Johnson, 2002) and (Bonnema and Scha, 2003). Figure 2.14 (adapted from the example given in (Bonnema and Scha, 2003):27) illustrates how this bias comes into play even for very small treebanks. The treebank trees given in Figure 2.14(A) yield the set of fragment in (B); trees T_a, T_b and T_c all yield fragment f_1 while tree T_d yields fragments f_2 - f_7. The probabilities associated with these fragments are obtained using the relative frequency estimator, meaning that each fragment probability is calculated by dividing its frequency of occurrence in the fragment base by the total number of fragments with the same root node as it. Figure 2.14(C) shows the five DOP derivations (and their probabilities) which can be constructed for the string *ab* given fragment set (B). Figure 2.14(D) shows, thus, that the DOP probability model (which sums over derivation probabilities) ranks T_{bigger}, the larger tree representing *ab*, higher than the smaller tree $T_{smaller}$. In other words, the DOP probability model considers that T_{bigger} is the more likely analysis for the string *ab*. This, however, is in contradiction of the evidence present in the original treebank in (A); treebank (A) predicts that the smaller parse tree for *ab* is three times more likely than the larger analysis. Consequently, we conclude that assigning probabilities to subtrees in terms of their relative frequencies results in an undesirable bias towards larger parse trees. Furthermore, Johnson (2002) shows that the estimation method is also inconsistent, i.e. that the estimated probability distribution does not converge to the true distribution as the size of the treebank increases.

[20] Collins and Duffy (2001) describe how the use of of kernel methods over trees can be applied to parsing. They assume all tree fragments which occur in the training data, as does DOP. However, unlike for DOP, enumeration of these fragments is conceptual rather than explicit. Instead, they discuss how kernels can be used to represent parse trees such that all subtrees are tracked, and how these kernels can be applied to parsing using the voted perceptron algorithm. The methods they propose allow for distribution-free parameter estimation techniques which can be applied efficiently. Although interesting, further discussion of these methods is beyond the scope of this thesis.

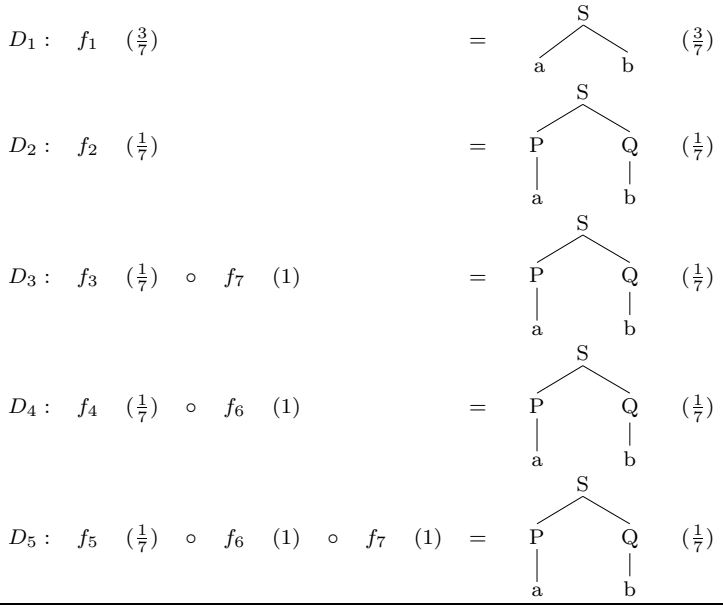
(**A**) **A sample treebank** :



(**B**) **The fragment set extracted from treebank** (**A**) **where fragment probabilties are calculated according to the relative frequency estimator** :



(**C**) **All derivations and derivation probabilities for the string** ab **according to treebank** (**B**) :



(**D**) **Parse probabilities for the string** ab **according to the DOP probability model and according to the evidence in the treebank** :

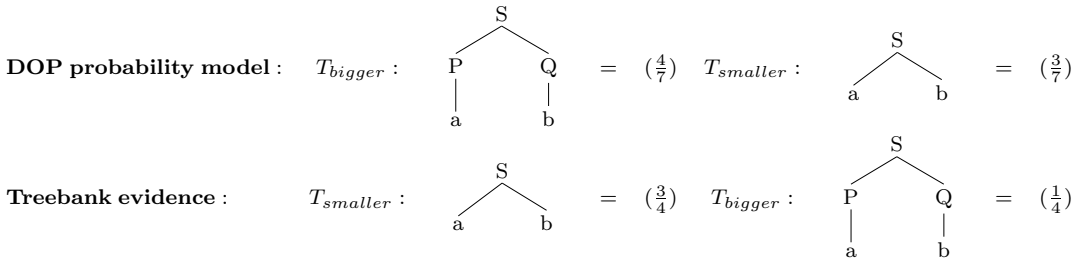


Figure 2.14: This example illustrates the bias in favour of larger parse trees induced by the DOP relative frequency estimator. This illustration is adapted from the example given in (Bonnema and Scha, 2003):27.

Bonnema and Scha (2003) note that, despite these demonstrable flaws in the way in which fragment probabilities are estimated, experimental results indicate that the DOP_{rf} model achieves very high parse scores. Furthermore, the vast majority of those experiments which investigate the effects of increasing fragment depth on parse accuracy validate the DOP Hypothesis, which states that accuracy is expected to increase as fragment size increases. These experiments are carried out using fragments of depth n or less such that the value of n is varied while the training and test data remain the same. That they validate the DOP Hypothesis is surprising in light of the DOP_{rf} bias presented above. However, closer examination of the experiments performed reveal that these experiments generally fall into one of two categories: those which used only a subset of the available fragments at each depth, e.g. (Sima'an, 1999; Bod, 2000d,e, 2001, 2003b), and those which were performed on the ATIS corpus, e.g. (Sima'an, 1995a; Bod, 1995b, 1996, 2000e, 2003c)).[21] The exception to this is the set of experiments performed by Bod (2000e) on the OVIS corpus, which comprises 10,000 trees containing both syntactic and semantic annotations. These experiments show that maximum parse accuracy is achieved at fragment depth 4 and including fragments of depths 5 and 6 results in decreased accuracy. However, Bod (2000e) attributes this finding to the fact that DOP suffers considerably from sparse data problems on the OVIS because the syntactic and semantic annotations at each node are treated as one label (Sima'an, 1999). Thus, the experiments which confirm that the DOP Hypothesis holds are generally either restricted in terms of the proportion of fragments actually used at each depth or performed on a corpus which yields a relatively low number of fragments at each depth. Consequently, the biased parameter estimator does not incur the negative impact on results that one might have expected to see.

2.6.2 Assuming uniform distribution over the training trees: DOP_{bon}

The solution proposed in (Bonnema et al., 2000; Bonnema and Scha, 2003) is based on the assumption that, as all corpus trees are formed by derivation sequences but we do not know which ones, all derivations of each corpus tree are equally likely. Furthermore, this

[21] The number of fragments which can be extracted from the ATIS corpus appears to be relatively small. Furthermore, some experiments on the ATIS corpus were on tag rather than word sequences, further reducing the number of fragments extracted. Therefore, the bias in the probability model is likely to have minimal impact for experiments carried out on this data.

uniform distribution can be used to estimate the *expected relative frequency of substitution* of each subtree. It is shown (Bonnema et al., 2000; Bonnema and Scha, 2003) that for every possible fragment α of tree T with root node A, the fraction of derivations of T that start by substituting α is $2^{-N(\alpha)}$, where $N(\alpha)$ is the number of non-root, non-terminal nodes in α. Consequently, they define the probability of fragment α with root node A as the product of α's frequency of substitution in the treebank tree derivations and its relative frequency in the set of all fragments as specified in equation (2.30).

$$P_{bon}(\alpha) = 2^{-N(\alpha)} P_{rf}(\alpha) \tag{2.30}$$

While it is stated in (Bonnema et al., 2000) and (Bonnema and Scha, 2003) that this estimation method is unbiased and consistent, and performs as desired on toy treebanks, Sima'an and Buratto (2003) present experiments on the OVIS corpus which show that DOP_{bon} achieves lower parse accuracy than DOP_{rf}. These experiments also limit the size of the fragment set not only with respect to depth but also with respect to the numbers of terminals and substitution sites in each fragment. We have already seen that this favours the DOP_{rf} model; one wonders what impact it has on the DOP_{bon} model and, consequently, whether or not this was a fair assessment. Nevertheless, Sima'an and Buratto (2003) also provide an example which clearly demonstrates that this estimation method is biased, this time towards *smaller* parses, and conclude that the bias towards smaller parses exhibited by DOP_{bon} is more harmful than the bias towards larger parses exhibited by DOP_{rf}.

2.6.3 Using Maximum Likelihood Estimation: DOP_{mle}

A Maximum Likelihood Estimator assigns probabilities to grammar rules – or, in the case of DOP, fragments – such that the resulting probabilistic grammar, which can generate parse trees not in the treebank from which it was extracted, assigns maximum probabilities to those trees which occurred in the original treebank. As outlined by Bonnema and Scha (2003), however, this estimation method is not suitable for DOP because, as the treebank trees also occur as fragments in the fragment base, MLE assigns all probability mass to those fragments corresponding to treebank trees and zero probability mass to all other

fragments in the fragment base. Thus, DOP_{mle} over-learns the training data as it can only generate those parse trees which already exist in the treebank.

If the set of training fragments are pruned with respect to depth – usually the baseline pruning strategy – then all of the treebank trees are no longer also fragments in the fragment set. Consequently, the effects of over-learning are constrained, at least to some extent. Bod (2000c) compared the DOP_{rf} and DOP_{mle} models by performing word-graph disambiguation experiments on the OVIS corpus where the fragment base contained all fragments of depth 4 or less; results showed that the DOP_{mle} model achieved a word error rate which improved on the score achieved by the DOP_{rf} model by 2.8%. According to Bonnema and Scha (2003), this result is entirely dependent on pruning, and that the more fragments allowed, the more marked the effects of overfitting will become.

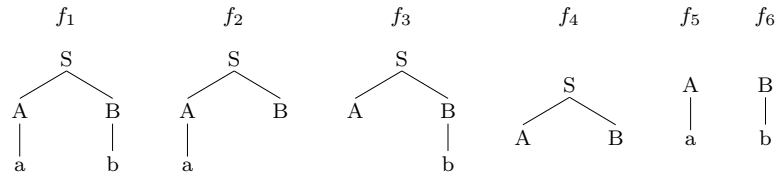
2.6.4 Probability re-estimation using Back-off: DOP_{bkf}

The models DOP_{rf}, DOP_{bon} and DOP_{mle} all view the DOP fragments as a set of disjoint events. Sima'an and Buratto (2003), on the other hand, view the DOP fragment set as a hierarchically structured space of correlated events and define a parameter estimation method which takes into account the relationships between overlapping fragments.

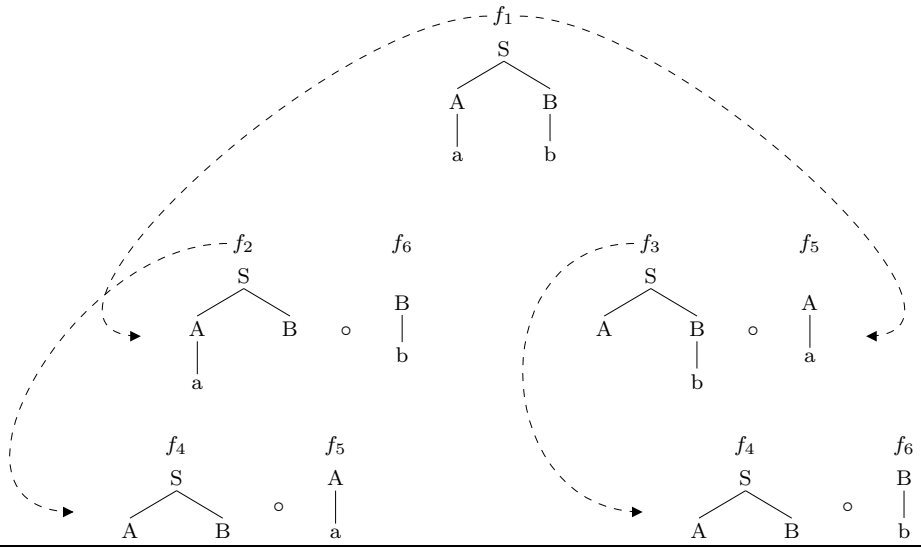
Sima'an and Buratto (2003) observe that all fragments in the fragment base of depth greater than 1 can be created by combining other fragments in the fragment base; depth 1 fragments are considered *simple* fragments and larger fragments are termed *complex* fragments. For example, fragment f_1 in Figure 2.15(A), which is a complex fragment, can be created by combining fragments f_2 and f_6 as shown in Figure 2.15(B). Fragment f_2 is also a complex fragment and can be created by combining fragments f_4 and f_5 - again, this is shown in Figure 2.15(B). Fragments f_4, f_5 and f_6 are all simple fragments, meaning that they cannot be broken down any further.

In DOP, the probability of f is given by $P(f|root(f))$. Thus, if $f = f_x \circ f_y$ then the probability of $f_x \circ f_y$ is given by $P(f_x \circ f_y|root(f_x))$. According to the chain rule, $P(f_x \circ f_y)$ is expanded according to equation (2.31). However, the DOP probability model allows

(**A**) **A set of DOP fragments** :



(**B**) **Organisation of the fragments in** (**A**) **as a structured space of correlated events** :



(**C**) **The backoff graph summarising the fragment relationships exposed in** (**B**) :

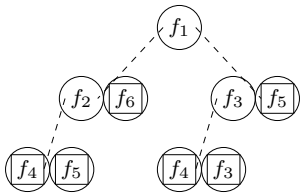


Figure 2.15: The fragments in (A) can be organised into the hierarchical structure in (B) by expressing complex fragments as compositions of complex and simple fragments. The back-off graph in (C) is a summary of this fragment hierarchy; complex fragments are enclosed by a circle and simple fragments (for which there are no back-offs) by a square and a circle.

the expansion of $P(f_x \circ f_y)$ according to equation (2.32).

$$P(f_x \circ f_y | root(f_x)) = P(f_x | root(f_x)) P(f_y | f_x) \tag{2.31}$$

$$P(f_x \circ f_y | root(f_x)) = P(f_x | root(f_x)) P(f_y | root(f_y)) \tag{2.32}$$

$$P(f_y | f_x) \approx P(f_y | root(f_y) \tag{2.33}$$

The only difference between equations (2.31) and (2.32) is in the conditioning context of f_y: in equation (2.31) it is conditioned on f_x whereas in equation (2.32) it is conditioned on $root(f_y)$ – this approximation is given in (2.33). As the root node of f_y corresponds to the leftmost substitution site of f_x, these are not unrelated. However, conditioning on $root(f_y)$ is weaker than conditioning on f_x as, essentially, it involves a 'backing-off' from conditioning on the entire of fragment f_x to just one node category in f_x. Hence, Sima'an and Buratto (2003) say that the derivation $f_x \circ f_y$ constitutes *a back-off* of subtree f. This relationship is expressed by equation (2.34).

$$f \geq_{bkf} f_x \circ f_y \tag{2.34}$$

The back-off relationships between fragments can be used to organise the fragment set into a hierarchically-structured space of correlated events as shown in Figure 2.15(B). Sima'an and Buratto (2003) sketch a graphical representation of this hierarchical space – called a *back-off graph* – using a directed acyclic graph where each node represents a pair of fragments $<f_x,f_y>$; a directed edge points from fragment f to each pair of fragments $<f_x,f_y>$ which derive it. The back-off graph corresponding to the organised fragment space in Figure 2.15(B) is given in Figure 2.15(C). Sima'an and Buratto (2003) firstly assign to each fragment its relative frequency and then use the back-off graph to re-estimate fragment probabilities by transferring probability mass from larger fragments to their back-offs in a stepwise fashion. DOP_{bkf} experiments carried out on the OVIS corpus show an error reduction of 11.3% over the DOP_{rf} method and 15.7% over DOP_{bon} (Sima'an and Buratto, 2003).

2.7 Summary

In this chapter, we have provided both general and formal descriptions of the DOP model, and discussed how the set of lexical and structural dependencies captured naturally using DOP differs from those captured by other approaches to experience-based parsing. We have also described methodologies which have been developed to address the issues of pruning, parsing and disambiguation for the DOP model. Finally, we have illustrated the unsatisfactory effects of using the relative frequency estimator to assign probabilities to DOP fragments and outlined three solutions which have been developed to address this problem.

In the next chapter, we present our Tree-DOP system in terms of design and performance. We detail our choice of algorithms to implement each of the main components in a DOP parser. We perform experiments on both English and French data, and compare our results to previously-published DOP results, as well as discussing in detail further characteristics of the DOP model which impact on performance.

Chapter 3

Tree-DOP: implementation, experiments and results

In this chapter, we document the Tree-DOP parser we have built in terms of design and performance. The system design reflects our overall motivation for building this parser: we require a Tree-DOP parser which is efficient, and yet flexible enough to serve as the main technology behind data-oriented language processing systems which assume tree-based representations encoding more information than simple context-free phrase-structure trees. Accordingly, in section 3.1 we motivate our choice of algorithms to accomplish the three main tasks performed by a Tree-DOP system, namely grammar induction, parsing and disambiguation. Of course, as this system is to form the core technology underlying further data-oriented language processing models, we wish also to confirm that it is accurate as well as sufficiently efficient and flexible. Thus, the first task when assessing performance – the details of which are presented in section 3.2 – is to compare the parse accuracy achieved using our system to the accuracy of other DOP systems (Bod and Kaplan, 2003) on the same data. We also compare some of the maximisation techniques which have been proposed for Tree-DOP in order to ascertain how they perform in terms of both accuracy and efficiency. In addition, we present detailed discussion on further characteristics of the DOP model which prove to impact on performance.

3.1 Parser design details

In this section, we present the precise description of our Tree-DOP system architecture in terms of grammar induction, parsing and disambiguation. We motivate our choice of algorithm for parse-space computation by comparing it to the other possible algorithms in section 3.1.1. We discuss the challenges which present themselves when creating, storing and compiling Tree-DOP grammars in section 3.1.2, and we give the algorithms we have selected to rank output parses in section 3.1.3.

3.1.1 Parse space computation

Three methods for computing the parse space for a given input string over a Tree-DOP grammar are described in section 2.4. The first of these methods, developed by Bod (e.g. (Bod, 1992, 1995a)), views each tree as a rewrite rule of the form $root(f) \longrightarrow frontier_1(f)\ ...\ frontier_n(f)$ and uses standard chart-parsing techniques to build the parse space for any given input string. This is the least efficient method as the number of rules to be considered is linear in the size of the fragment set and the right-hand sides of these rules are of arbitrary length where the maximum length is the length of the longest string in the training data. Furthermore, while use of standard chart-parsing techniques to parse with these rewrite rules is possible, it is also impractical: a bottom-up, CKY-based approach requires conversion of the rule set to CNF which causes an explosion in the number of rules, while a top-down approach, even incorporating lookahead techniques, necessitates the introduction of an undesirable number of failing derivations into the parse space. As the second and third methods – developed by Sima'an and Goodman – offer far greater efficiency, we do not consider this method any further.

The two-phase algorithm developed by Sima'an (1995a, 1999) and the PCFG-reduction developed by Goodman (1996a, 1998, 2003) are both predicated on the same underlying idea: all of the fragments which can be extracted from a treebank can also be generated by the context-free grammar underlying that treebank and, consequently, the context-free rules present in the parse-space for a given input string also characterise all of the fragments which can be used to parse that string. However, there are two fundamental differences between the two algorithms.

Firstly, Sima'an's algorithm uses the non-probabilistic context-free grammar underlying the fragments to compute exactly the set of fragments relevant to the parse space but not the probabilities of those fragments; these probabilities must be estimated by looking back to the full fragment set. In contrast, Goodman's method assigns probabilities to each underlying context-free rule such that the sum of the PCFG derivation probabilities yielding a particular fragment is exactly the DOP probability of that fragment in the fragment set.

The second fundamental difference between the two algorithms concerns the format of the parse spaces they generate; this is crucial as it determines which disambiguation techniques can be applied. Sima'an integrates the second phase of his algorithm (during which the context-free rules are used to establish the set of relevant fragments) with the Viterbi algorithm to calculate the most probable derivation for the input string. Consequently, only those fragments used to build the most probable derivation are actually introduced into the parse space. However, as his method can easily be extended so that the parse space generated comprises all relevant fragments, the fragments participating in the n most probable derivations, the fragments participating in the shortest derivation(s) etc., it can be used in conjunction with a range of ranking techniques. (The notable exception to this is Goodman's maximum constituents parse.) In contrast, the parse space generated using Goodman's algorithm does not contain fragments; rather it comprises PCFG rules which generate the same parses and parse probabilities as the DOP model for the given input string. Goodman proposes to disambiguate this chart by selecting the maximum constituent parse, which does not require conversion of the PCFG parse space to the DOP parse space. Unfortunately, integrating alternative ranking strategies with this algorithm is not straightforward. As more than one PCFG derivation can yield each DOP derivation, computation of the most probable PCFG derivation (and also, therefore, the n most probable derivations using Viterbi) does not guarantee that we have found the most probable DOP derivation. If there exists a unique shortest derivation then this can be generated, but where there is more than one shortest derivation (as is generally the case) then standard backing-off to the most probable derivation is, again, problematic.[1] Com-

[1] Bod (2003a) documents experiments whereby both the n shortest derivations and the n likeliest derivations are computed from the PCFG-reduction parse space using Viterbi optimisation. However, precise

putation of the most probable parse using random sampling involves selecting fragments for composition at random according to the probability distribution over the competition set. However, the sampling probability distributions over sets of rules do not necessarily correspond to the sampling probability distributions over sets of fragments, meaning that the distribution of randomly sampled derivations from the parse space generated by the Goodman algorithm is not guaranteed to correspond to the distribution according to the DOP model and, therefore, it is not clear how to identify the DOP most probable parse. Thus, use of maximisations other than the maximum constituent parse would appear to require the conversion from PCFG parse space to DOP parse space which the algorithm was developed to avoid.

Finally, use of pruning techniques to limit the size of the fragment space is straightforward using Sima'an's algorithm as the CFG rules in the underlying grammar are extracted directly from the set of fragments being used. On the contrary, it is not straightforward to estimate Goodman's CFG rule probabilities such that excluded fragments are adjusted for; Goodman (2003) describes a method to perform pruning with respect to fragment depth but a far larger PCFG must be extracted from the treebank to accomplish this.

In summary, in order to output parses other than the maximum constituents parse, we must convert Goodman's PCFG-reduction parse space to the DOP parse space. However, for a treebank even of reasonable size, the number of fragments in the parse space will be extremely large, and explicitly computing it is unfeasible in terms of both time and space. Pruning the fragment set so that the parse space is computable unfortunately results in a large increase in the size of the PCFG-reduction (if, indeed, it is even possible to compute the corresponding PCFG-reduction) and the Goodman algorithm loses its advantage. Consequently, the decision as to which approach to take when building a Tree-DOP parser depends on the degree of flexibility required. If selecting the maximum constituents parse from the parse space generated using the full set of DOP fragments yielded by the given treebank is appropriate for the task at hand, then Goodman's algorithm is most suitable. If, however, use of pruning techniques and/or alternative ranking strategies is necessary then Sima'an's algorithm would appear to provide the better solution.[2] As we wish to

details of how the issues raised here are addressed are not given.

[2]Flexibility is also required if data-oriented language processing models involving different types of

investigate precisely these issues, we have adopted this algorithm as it was described in section 2.4.2.

3.1.2 Compact fragment representation

The task of deriving a DOP grammar from a treebank by extracting sets of fragments from the treebank and computing their frequencies, as well as storing and compiling these grammars, is computationally expensive. Fortunately, as the two-phase algorithm used to compute the parse space for each input string requires only an indication as to which fragments each underlying CFG rule appears in, it is not necessary to explicitly extract and store the fragment set. Rather, we store only the treebank trees themselves and establish the fragment set on the fly.

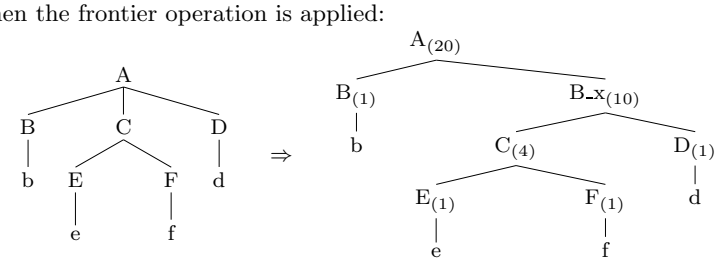
This is accomplished by first explicitly applying the root operation to the treebank trees, yielding a set of 'intermediate' fragments the size of which is linear in the number of nodes in the treebank. The frontier operation is then applied by assigning to each node n in each intermediate fragment a set of fragment identifiers such that if its left and right child nodes n_l and n_r are present in a fragment, then the corresponding fragment identifier appears in the node's identifier set. Either both n_l and n_r are present in the fragment or neither are present, in which case node n is itself either a substitution site or not in the fragment. Thus, the presence of fragment identifier f_{id} at node n signifies that the CFG rule $n \longrightarrow n_l\ n_r$ occurs in fragment f_{id}.

If n's left and right child nodes n_l and n_r are present in a fragment, each of these child nodes can be either internal to that fragment (n_{l_i},n_{r_i}) or a substitution site of that fragment (n_{l_s},n_{r_s}). Thus, we can partition the set of identifiers at node n into four sets representing the four possible combinations of internal and external child nodes $<n_{l_s},n_{r_s}>$, $<n_{l_s},n_{r_i}>$, $<n_{l_i},n_{r_s}>$ and $<n_{l_i},n_{r_i}>$.[3] Extracting these partitioned sets of fragment identifiers along with each CFG rule extracted gives us the correspondence between the fragment set and the CFG underlying it which is required to perform the transition from

representations are to be constructed. Examples in this thesis are the paired tree representations used for translation and the trees associated with f-structures for LFG parsing. The relative merits of each algorithm for these types of representations are discussed in the relevant chapters.

[3] We can also partition according to whether n is a root or internal node, creating eight partitions rather than four; for the sake of clarity, we omit this option in this explanation.

(A) Root-generated 'intermediate' fragment which has been converted to ECNF (through which node B_x has been inserted) and each node annotated with the number of different subtrees it yields when the frontier operation is applied:



(B) Node annotations representing all possible frontier operations where the total number of frontier operations possible is 20 and the fragments corresponding to each of these frontier operations have been allocated identifiers from the set of integers 1 - 20:

$A_{(20)}$	<B_s,B_x_s>:{} <B_s,B_x_i>:{1-10} <B_i,B_x_s>:{} <B_i,B_x_i>:{11-20}
$B\_x_{(10)}$	<C_s,D_s>:{1,11} <C_s,D_i>:{2,12} <C_i,D_s>:{3-6,13-16} <C_i,D_i>:{7-10,17-20}
$C_{(4)}$	<E_s,F_s>:{3,7,13,17} <E_s,F_i>:{4,8,14,18} <E_i,F_s>:{5,9,15,19} <E_i,F_i>:{6,10,16,20}
$B_{(1)}$	<b>:{11-20}
$E_{(1)}$	<e>:{5-6,9-10,15-16,19-20}
$F_{(1)}$	<f>:{4,6,8,10,14,16,18,20}
$D_{(1)}$	<d>:{2,7-10,12,17-20}

Figure 3.1: The 'intermediate' fragment in (A) was generated by the root operation. (B) gives the node annotations representing all possible frontier operations where the total number of frontier operations possible is 20 and the fragments corresponding to each of these frontier operations have been allocated identifiers from the set of integers 1 - 20.

parse phase 1, in which the CFG parse space is constructed, to parse phase 2 in which the corresponding DOP parse space is constructed.

The process of building compact fragment representations is illustrated in Figure 3.1. Firstly, node A is selected by the root operation and all nodes not equal to or dominated by A are deleted; this yields the 'intermediate' fragment given in Figure 3.1(A). This intermediate fragment is converted to ECNF as described in section 2.4.4 through the insertion of the new node B_x, and the number of frontier operations which can be carried out at each of its nodes is calculated. For example, 10 different sets of frontier nodes can be selected at node B_x; note, however, that as B_x must always be an internal node and, therefore, never a substitution site, the number of different frontier node sets

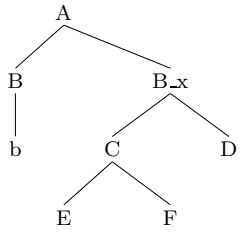
which can be selected at node A is 20.[4] Fragment identifiers are then assigned to each node in the intermediate tree as shown in Figure 3.1(B): an identifier appears at a given node if both that node and all of its child nodes appear in the corresponding fragment. These sets of identifiers are further partitioned as described above. For example, the sets corresponding to node A indicate that node B_x is internal to all fragments 1–20 but node B is a substitution site in fragments 1–10 and internal to fragments 11-20. Similarly, the sets corresponding to node B_x indicate that nodes C and D are, for example, both substitution sites in fragments 1 and 11 (where node B is a substitution site in 1 and internal to 11) and both are internal to fragments 7–10 and 17–20 (where node B is a substitution site in 7–10 and internal to 17–20). As terminal symbols can only be frontier nodes, they are either present or absent in each fragment and, thus, no partitions are imposed on their fragment identifier sets. For example, terminal symbol b is internal to all fragments to which its parent node B is also internal, i.e. fragments 11-20.

This method of representing fragments allows us, when extracting the CFG underlying the treebank, to also extract for each rule the (partitioned sets of) identifiers of fragments in which that rule occurs. This annotated grammar can be used during the two-phase analysis process to (i) establish the CFG parse space for the input string and (ii) transition to the DOP parse space for that string as described in section 2.4.2. In addition, it allows us to read off the fragment corresponding to any identifier by simply checking for its absence or presence (as an internal node or substitution site) at each node in the intermediate tree.[5] For example, consider the situation where we wish to extract the fragment whose identifier is 13. The sets corresponding to node A indicate that nodes B and B_x are both internal to fragment 13. Trivially, the annotation at node B indicates that the terminal b is a frontier node of fragment 13. The sets corresponding to node B_x indicate that while node C is internal to fragment 13, node D is a substitution site. Finally, the sets corresponding to node C tell us that nodes E and F are both substitution sites in fragment

[4] If node B_x was allowed to be a substitution site, two further fragments would be possible; both these fragments would have node B_x as a substitution site but one would also have node B as a substitution site whereas B would be internal to the other one.

[5] Sima'an's two-phase parsing algorithm does not require this facility as fragments are rebuilt using the CFG rules which characterise them. However, this facet of the compact fragment representation process will prove important when tree-based representations encoding more information than simple context-free phrase-structure trees are considered. This issue is discussed further in sections 5.2.3, 7.5.2 and 8.3.

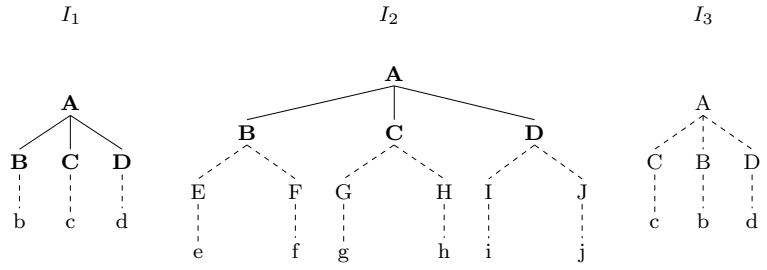
13. Thus, fragment 13 corresponds to the fragment shown in (3.1):



(3.1)

Calculating relative frequencies from compact fragment representations

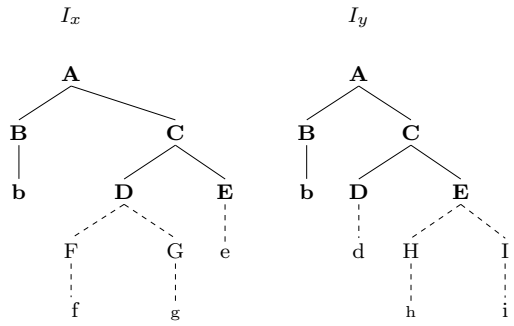
Calculation of relative frequencies (and the removal of identifiers corresponding to duplicate fragments) over these compact fragment representations is straightforward. Two intermediate trees I_x and I_y encode duplicate DOP fragments if connected portions of those trees which start at their root nodes are identical. Minimally, these connected portions must comprise the intermediate tree root nodes and their daughter nodes. Additionally, for two minimal portions to be identical, all node categories must be the same and, in the case of the daughter nodes, appear in the same order. In example (3.2), we see that intermediate trees I_1 and I_2 have the same minimal portions, i.e. their root nodes are of the same category and the children of those root nodes are of the same categories and in the same order. In contrast, the minimal portion of tree I_3 has the same root node category and daughter node categories as I_1 and I_2 but those daughter node categories do not match with respect to order and so the depth 1 fragment extracted from I_3 is not the same as the depth 1 fragment extracted from both I_1 and I_2.



(3.2)

Extending the portions of intermediate trees which yield identical DOP fragments is a recursive process: for each node already in the identical portion of each tree, we simply check that all of its daughter nodes correspond to those of its identical counterpart in the

intermediate tree to which it is being compared. In example (3.2), the identical portions of I_1 and I_2 can be extended no further as none the daughter nodes of B, C and D in I_1 match the corresponding daughter nodes in I_2.[6] In contrast, in example (3.3) we see that intermediate trees I_x and I_y have root node category A, and A's daughter nodes in both trees are (from left to right) B and C. Accordingly, those depth 1 fragments with root node A and substitution sites B and C are duplicates of each other. In addition, the identical portions of I_x and I_y can be extended to include nodes D and E as the daughter nodes of C also correspond. However, the identical portions can be extended no further as the children of D and E do not correspond.



(3.3)

Once we have identified the tree nodes included in the identical tree portions, we have established exactly which fragments are duplicates of each other: all boundary identical nodes (i.e. those nodes which are included in the identical tree portions but whose children are not) are either substitution sites of those fragments which are duplicates, or not contained in duplicate fragments. When we have identified these fragments, we simply increment their counts in one intermediate tree and delete their identifiers from the other.

3.1.3 Ranking parses

As discussed, use of two-phase analysis allows for flexibility as to the disambiguation strategy to be used. Consequently, we have built modules which compute the most probable parse, the most probable derivation and the shortest derivation.

In order to compute the most probable parse, the full DOP parse space is computed during analysis and the methods of Chappelier and Rajman (2003) direct the sampling

[6] Identical tree portions are depicted using bold type and solid lines, and non-identical portions using dashed lines.

process. More specifically, computation of exact sampling probabilities allows control over the number of samples taken, as described in section 2.5.1.

The Viterbi algorithm is integrated with the building of the parse space by including only the fragment with the largest inside probability for each root node category at each chart position, thus facilitating computation of the most probable derivation.

Calculation of the shortest derivation is also integrated with the computation of the parse space using Viterbi by adopting Bod (2000e)'s strategy of assigning all fragments equal probability $\frac{1}{p}$ and calculating all shortest derivations. However, in the event that there exists more than one shortest derivation, we select the most probable derivation amongst the n shortest derivations by performing a second pass over the parse space using the Viterbi algorithm and the DOP probabilities for each fragment.

3.2 Experiments and results

In section 3.2.1, we give details of the experiments we carried out in terms of the data employed and languages covered, the pruning and ranking strategies used, and the handling of input not fully covered by the fragment set. We describe how each evaluation metric we use is calculated in section 3.2.2 and present the results of our experiments in terms of the accuracy of the output parses according to these metrics in sections 3.2.3 and 3.2.4. Finally, in section 3.2.5 we discuss the issues which arise from these results and draw some conclusions.

3.2.1 Experimental set-up

We present Tree-DOP parsing experiments on the English and French sections of the Xerox HomeCentre corpus. This corpus comprises 980 English sentences and 930 French sentences from a printer manual – although the English and French corpora are translations of each other, we consider the sets of sentences independently for the purposes of these experiments.[7] On average, there are 9.20 English words per sentence and 10.82 French words per sentence. The longest English sentence comprises 34 words and the

[7]The English and French corpora are translations of each other, but there is not always a 1-to-1 mapping between sentences; this explains why the corpus sizes differ.

longest French sentence 39 words. The average depth of the English trees is 5.68 and the average depth of the French trees is 6.23; the maximum depth of both the English and French trees is 17. Each sentence is annotated with an LFG representation comprising a phrase-structure tree, an attribute-value matrix and links between them; we discuss these representations further in chapters 7 and 8. For these experiments, we extracted only the phrase-structure trees corresponding to each sentence. We preprocessed these trees by removing traces and empty categories and by removing unary-branching structures, i.e. substructures of the form $X \longrightarrow Y$ were replaced with the Y category. These changes were made fully automatically and, therefore, in a consistent manner; no manual alterations of any kind were made to the data. Finally, each dataset was split randomly into 8 training/test splits such that all test words also appeared in the training set. The English splits contain 90 test sentences and 890 training trees each while the French splits contain 90 test sentences and 840 training trees each.

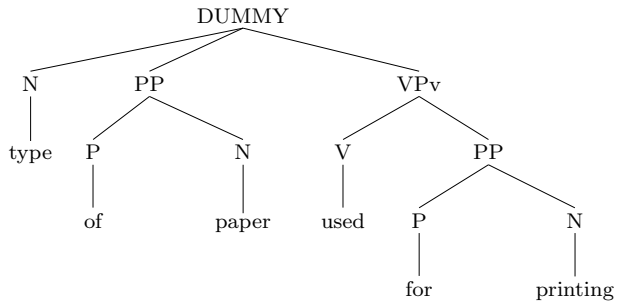
We parse each test sentence[8] using the three ranking strategies – most probable parse,[9] most probable derivation and shortest derivation (referred to as MPP, MPD and SDer) – as they are described in section 3.1.3 above. Furthermore, we also prune the fragment base extracted from each training set with respect to depth, resulting in fragment bases comprising fragments of depth 1, depth 2 or less, depth 3 or less and depth 4 or less. As there are 12 ways of combining the ranking and pruning strategies, each test sentence in each split is parsed in 12 different ways and the accuracy of the parses obtained are averaged over all splits for each combination.

It is not always the case that every test sentence in every split will be assigned a full parse. Where a sentence does not receive a full parse, we assign to that sentence the best sequence of partial parses according to the relevant ranking strategy. We combine these partial parses into a single tree by simply inserting a fake root node with category DUMMY such that all the partial parses are siblings and their parent node is the fake node. (In the worst case, every word in the sentence is assigned the tag DUMMY and all such DUMMY

[8] All experiments are carried out on a Pentium 4 with 2.39GHz CPU and 2Gb RAM.

[9] When computing the most probable parse for each input string using random sampling, we set the sampling thresholds P_{err} and θ described in section 2.5.1 to 0.01 and 2 respectively. We also set the maximum number of samples to 10,000 so that, in the event of there being two or more equally likely parses, sampling will terminate. For the parsing experiments we present, this situation never arose, meaning that we could always distinguish one parse as being more probable than the others.

$\longrightarrow$ *word* subtrees are considered siblings with parent node DUMMY; no sentences in our experiments fell into this 'worst case' category.) This is illustrated by the trivial example in (3.4) below, where three partial parses are combined using category DUMMY. Thus, every sentence receives the best analysis which can be assigned to it, even if that analysis is incomplete.



(3.4)

3.2.2 Evaluation metrics

The output parse for each test sentence is evaluated by comparing it to the parse which was assigned to that test sentence in the corpus – this parse was stripped off when the corpus was split into training and test sets but held out as a reference parse to be used for evaluation purposes. We evaluate parses against their reference parses using 4 metrics: exact match, precision, recall and f-score. The exact match metric is the most stringent, in that parses which are identical to their reference parses are assigned a score of 1 and all others assigned a score of 0. The precision, recall and f-score metrics, on the other hand, compare the constituents present in the output parse with those present in the reference parse, where a constituent is a syntactic category label occurring in a parse tree which spans a consecutive sequence of words in the input string. A constituent is correct if there is a corresponding node in the reference parse with the same syntactic category label spanning the same consecutive sequence of input string words. Precision is calculated according to equation (3.5), where P is the parser output and T the reference parse. The precision rate of a parse is the proportion of constituents in that parse which are correct.

$$Precision = \frac{\#\ correct\ constituents\ in\ P}{\#\ total\ number\ of\ constituents\ in\ P} \tag{3.5}$$

Recall is calculated according to equation (3.6); the recall rate of a parse is the proportion of correct constituents in the parse with respect to the total number of constituents in the

reference parse.

$$Recall = \frac{\#\ correct\ constituents\ in\ P}{\#\ total\ number\ of\ constituents\ in\ T} \tag{3.6}$$

F-score is a method of combining precision and recall to facilitate comparison and is calculated according to equation (3.7).

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.7}$$

In PCFG parsing, the tasks of tagging and parsing are generally considered separately; sentences are tagged using a tagger and the tag sequences input to the parser. Consequently, tagging and parsing are evaluated separately, meaning that pre-terminals spanning a single word (i.e. structures of the form TAG $\longrightarrow$ terminal) are not considered constituents for parser evaluation purposes. Input to our DOP parser, on the other hand, comprises sequences of terminal symbols *without* their tags, meaning that the parser assigns both tags and structures to each input string. Accordingly, structures of the form TAG $\longrightarrow$ terminal *are* treated as constituents during parser evaluation, and the precision, recall and f-score figures we report also incorporate tagging accuracy. However, we also perform a separate evaluation of tagging accuracy by calculating the percentage of words in each sentence which are correctly tagged.

3.2.3 Results for English experiments

The results given in Table 3.1 – which are calculated over all parses produced, be they complete or partial – demonstrate the effects on parse accuracy, for each ranking strategy, of increasing the size of the fragment base by including fragments of greater depth. Results for the MPP ranking indicate that accuracy increases according to all metrics as fragment depth increases from depth 1 to depth 2. However, only the exact match figure improves as fragment depth is increased to 3; precision, recall and f-score all show reduced accuracy as depth 3 fragments are included. Furthermore, all metrics show a deterioration in output quality as depth 4 fragments are included. Thus, parse accuracy peaks at depth 2 in terms of precision, recall and f-score and at depth 3 according to exact match when the MPP ranking is used. Results for the output parses ranked according to the MPD show improved

ALL PARSES

	Most Probable Parse (MPP)				
	precision	recall	f-score	exact	tags
1	91.89	90.48	91.18	68.89	97.01
2	**94.75**	**93.65**	**94.20**	78.89	**98.34**
3	94.64	93.46	94.05	**80.00**	98.01
4	92.70	91.63	92.17	68.89	97.68
	Most Probable Derivation (MPD)				
	precision	recall	f-score	exact	tags
1	91.44	91.44	91.44	62.22	97.68
2	**94.17**	**93.17**	**93.67**	**77.78**	**98.18**
3	93.48	92.40	92.94	75.56	97.51
4	92.23	91.25	91.74	65.56	97.51
	Shortest Derivation (SDer)				
	precision	recall	f-score	exact	tags
1	91.15	89.13	90.13	65.56	96.35
2	93.67	92.50	93.08	76.67	97.84
3	**93.77**	**92.60**	**93.18**	**77.78**	**98.01**
4	93.20	92.21	92.70	68.89	97.84

Table 3.1: ENGLISH. Results demonstrating the effects on parse accuracy, for each ranking strategy, of increasing the size of the fragment base by including fragments of greater depth.

ALL PARSES

	F-Score				Exact Match		
	MPP	MPD	SDer		MPP	MPD	SDer
1	91.18	**91.44**	90.13	1	**68.89**	62.22	65.56
2	**94.20**	93.67	93.08	2	**78.89**	77.78	76.67
3	**94.05**	92.94	93.18	3	**80.00**	75.56	77.78
4	92.17	91.74	**92.70**	4	**68.89**	65.56	**68.89**

Table 3.2: ENGLISH. Results showing the relative performance of ranking strategies MPP, MPD and SDer as fragment depth increases.

accuracy across all metrics as fragment depth increases from 1 to 2. However, enlarging the fragment base to include fragments of depths 3, and then fragments of depth 4, results in subsequent decreases in output quality. Thus, parse accuracy for MPD ranking peaks at depth 2 according to all metrics. The quality of the output parses improves across all metrics as the fragment base is enlarged to include fragments of depth 2 and then fragments of depth 3 when the SDer ranking method is used. Accuracy decreases, however, when fragments of depth 4 are included. Thus, we observe that parse accuracy peaks at depth 3 according to all metrics for the SDer ranking strategy.

The results given in Table 3.2 – which are, again, calculated over both partial and complete parses – show the relative performance of the three ranking strategies as fragment depth increases. At depth 1, the highest f-score is achieved when parses are ranked

according to the MPD whereas the highest percentage of exact matches are achieved using the MPP rankings. At depths 2 and 3, both the highest f-score figures and the highest exact match figures are achieved when parses are ranked according to the MPP. However, at depth 4 the highest f-score figures shown correspond to the parses ranked using SDer, and the SDer and MPP strategies both beat the MPD rankings on the exact match metric where they achieve exactly the same score. Despite the fact that SDer outperforms the MPP and MPD rankings at depth 4, overall best performance is achieved using MPP rankings at depths 2 and 3: the overall best f-score is achieved using MPP ranking at depth 2 and the best exact match score is achieved using MPP at depth 3.

The results presented thus far have been calculated over both partial and complete parses. On average, 90.83% of sentences in each test set are assigned full parses according to the training data and 9.17% receive only partial parses. In order to investigate whether the trends observed over all parses are also in evidence when we distinguish between partial and complete parses, we present separate evaluations of the quality of complete and partial output in Tables 3.3 and 3.4; in each of these tables, the results in column (A) (on the left) refer to evaluation over full parses only and the results in column (B) (to the right) refer to evaluation over partial parses only.

We look first at the trends observed from the results presented in Table 3.1. These show that parse accuracy peaks at depth 2 in terms of precision, recall and f-score and at depth 3 according to exact match for MPP ranking, and that parse accuracy for MPD ranking peaks at depth 2 according to all metrics and at depth 3 for SDer ranking for all metrics. Column (A) of Table 3.3, which gives evaluations over full parses only, shows that while the same trends are observed for MPD and SDer, MPP ranking now shows peak performance at depth 3 for the precision, recall and f-score metrics as well as exact match. Conversely, column (B) of Table 3.3, which gives evaluations over partial parses only, shows peak f-scores for MPP ranking at depth 2 as before whereas peak MPD performance is observed at depth 1 and peak SDer performance at depth 2. Note that exact match scores for partial parses are always 0 as it is not possible for a partial parse to be identical to its reference parse.

Secondly, we look at the trends observed from the results presented in Table 3.2 which

(A) FULL PARSES ONLY (90.83%)

	Most Probable Parse (MPP)				
	precision	recall	f-score	exact	tags
1	94.95	94.63	94.79	76.54	98.43
2	97.83	97.83	97.83	87.65	**99.61**
3	**98.17**	**98.17**	**98.17**	**88.89**	**99.61**
4	95.78	95.89	95.83	76.54	99.02
	Most Probable Derivation (MPD)				
	precision	recall	f-score	exact	tags
1	94.14	95.43	94.78	69.14	99.02
2	**97.38**	**97.71**	**97.55**	**86.42**	**99.80**
3	96.80	96.91	96.86	83.95	99.02
4	95.32	95.54	95.43	72.84	99.02
	Shortest Derivation (SDer)				
	precision	recall	f-score	exact	tags
1	93.76	92.8	93.28	72.84	97.45
2	96.58	96.69	96.63	85.19	99.21
3	**97.14**	**97.14**	**97.14**	**86.42**	**99.61**
4	96.47	96.69	96.58	76.54	99.41

(B) PARTIAL PARSES ONLY (9.17%)

	Most Probable Parse (MPP)				
	precision	recall	f-score	exact	tags
1	74.34	68.48	71.29	0	89.36
2	**77.12**	**71.52**	**74.21**	0	**91.49**
3	74.34	68.48	71.29	0	89.36
4	75.00	69.09	71.92	0	90.43
	Most Probable Derivation (MPD)				
	precision	recall	f-score	exact	tags
1	**75.82**	**70.30**	**72.96**	0	**90.43**
2	75.50	69.09	72.15	0	89.36
3	74.34	68.48	71.29	0	89.36
4	74.34	68.48	71.29	0	89.36
	Shortest Derivation (SDer)				
	precision	recall	f-score	exact	tags
1	76.16	69.7	72.78	0	**90.43**
2	**76.82**	**70.30**	**73.42**	0	**90.43**
3	74.34	68.48	71.29	0	89.36
4	74.34	68.48	71.29	0	89.36

Table 3.3: ENGLISH. Results in column (A) show the effect on the parse accuracy of sentences which received full parses, for each ranking strategy, of increasing fragment depth. Results in column (B) show the effect on the parse accuracy of sentences which were not assigned full parses, for each ranking strategy, of increasing fragment depth.

(A) FULL PARSES ONLY (90.83%)

	F-Score		
	MPP	MPD	SDer
1	**94.79**	94.78	93.28
2	**97.83**	97.55	96.63
3	**98.17**	96.86	97.14
4	95.83	95.43	**96.58**

	Exact Match		
	MPP	MPD	SDer
1	**76.54**	69.14	72.84
2	**87.65**	86.42	85.19
3	**88.89**	83.95	86.42
4	**76.54**	72.84	**76.54**

(B) PARTIAL PARSES ONLY (9.17%)

	F-Score		
	MPP	MPD	SDer
1	71.29	**72.96**	72.78
2	**74.21**	72.15	73.42
3	**71.29**	**71.29**	**71.29**
4	**71.92**	71.29	71.29

	Exact Match		
	MPP	MPD	SDer
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0

Table 3.4: ENGLISH. Results in column (A) show the relative performance of ranking strategies MPP, MPD and SDer as fragment depth increases on sentences which received full parses. Results in column (B) show the relative performance of ranking strategies MPP, MPD and SDer as fragment depth increases on sentences which were not assigned full parses.

ALL PARSES

	Most Probable Parse (MPP)				
	precision	recall	f-score	exact	tags
1	89.74	90.22	89.98	52.22	97.68
2	92.93	93.55	93.24	64.44	98.72
3	**93.92**	**94.54**	**94.23**	**72.22**	98.84
4	93.53	94.21	93.87	70.00	**98.95**
	Most Probable Derivation (MPD)				
	precision	recall	f-score	exact	tags
1	90.03	90.75	90.39	50.00	98.03
2	93.36	93.61	93.49	**68.89**	**98.95**
3	93.18	93.68	93.43	66.67	98.72
4	**93.52**	**94.15**	**93.83**	66.67	**98.95**
	Shortest Derivation (SDer)				
	precision	recall	f-score	exact	tags
1	88.29	86.29	87.28	47.78	97.44
2	91.12	90.75	90.93	62.22	98.84
3	92.78	93.21	93.00	**66.67**	**98.95**
4	**92.99**	**93.55**	**93.27**	**66.67**	**98.95**

Table 3.5: FRENCH. Results demonstrating the effects on parse accuracy, for each ranking strategy, of increasing the size of the fragment base by including fragments of greater depth.

show that the best f-scores were obtained using MPD at depth 1, MPP at depths 2 and 3 and SDer at depth 4 and that the best f-score overall is achieved using MPP ranking at depth 2. The f-scores given in column (A) of Table 3.4 yield similar observations: the best f-scores were obtained using MPP at depths 2 and 3 and SDer at depth 4. The best f-score overall is achieved using MPP at depth 2 but the best f-score at depth 1 was this time obtained using MPP. Conversely, the f-scores given in column (B) of Table 3.4 yield more contradictory observations: the best f-scores were obtained using MPD at depth 1 and MPP at depth 2, as for all parses, but all three ranking strategies achieved the same f-scores at depth 3 and MPP ranking outperformed SDer ranking at depth 4. Thus, the results over full parses only show similar patterns to those seen over all parses whereas the partial parse results are less predictable. This is clearly a desirable balance as full parses are far more likely to be reliable than partial ones. Consequently, we observe that evaluating complete and partial parses together does not unduly skew the results.

3.2.4 Results for French experiments

The results given in Table 3.5 – which are calculated over all parses produced, be they complete or partial – demonstrate the effects on parse accuracy, for each ranking strategy, of increasing the size of the fragment base by including fragments of greater depth. Results

ALL PARSES

	F-Score				Exact Match		
	MPP	MPD	SDer		MPP	MPD	SDer
1	89.98	**90.39**	87.28	1	**52.22**	50.00	47.78
2	93.24	**93.49**	90.93	2	64.44	**68.89**	62.22
3	**94.23**	93.43	93.00	3	**72.22**	66.67	66.67
4	**93.87**	93.83	93.27	4	**70.00**	66.67	66.67

Table 3.6: FRENCH. Results showing the relative performance of ranking strategies MPP, MPD and SDer as fragment depth increases.

for the MPP ranking indicate that accuracy increases according to all metrics as fragment depth increases from depth 1 – 3. However, only the tagging accuracy figure improves as fragment depth is increased to 4; precision, recall and f-score all show reduced accuracy as depth 4 fragments are included. Thus, parse accuracy is highest at depth 3 in terms of precision, recall, f-score and exact match when the MPP ranking is used. Results for the output parses ranked according to the MPD show improved accuracy for the precision, recall and f-score metrics at each increase in fragment depth from 1 to 4. However, exact match accuracy is highest at depth 2; accuracy decreased by 2.22% as fragments of depth 3 were included and then remained the same when fragments of depth 4 were introduced. Thus parse accuracy when calculating the MPD peaks at depth 4 for the precision, recall and f-score metrics and at depth 2 for the exact match metric. No deterioration in the quality of the output parses is seen for any metric as the fragment base is enlarged when the SDer ranking method is used, although exact match and tagging accuracy do not improve after depth 3. Thus, we observe that parse accuracy is highest at depth 3 for the exact match metric and at depth 4 according to the precision, recall and f-score metrics for the SDer ranking strategy.

The results given in Table 3.6 – again calculated over both partial and complete parses – show the relative performance of the three ranking strategies as fragment depth increases. At depths 1 and 2, the highest f-score is achieved when parses are ranked according to the MPD whereas the highest percentage of exact matches is achieved using the MPP ranking at depth 1 but the MPD ranking at depth 2. At depths 3 and 4, both the highest f-score figures and the highest exact match figures are achieved when parses are ranked according to the MPP. SDer ranking never outperforms MPD ranking in terms of f-score but matches it in terms of exact match scores at depths 3 and 4. Overall, best performance is achieved

(A) FULL PARSES ONLY (92.36%)

	Most Probable Parse (MPP)				
	precision	recall	f-score	exact	tags
1	92.12	92.32	92.22	55.29	98.18
2	95.92	96.27	96.10	68.24	**99.48**
3	**96.88**	**97.24**	**97.06**	**76.47**	**99.48**
4	96.44	96.87	96.65	74.12	**99.48**
	Most Probable Derivation (MPD)				
	precision	recall	f-score	exact	tags
1	92.44	92.92	92.68	52.94	98.57
2	96.13	96.13	96.13	**72.94**	**99.48**
3	95.99	96.20	96.09	70.59	99.22
4	**96.44**	**96.80**	**96.62**	70.59	**99.48**
	Shortest Derivation (SDer)				
	precision	recall	f-score	exact	tags
1	90.44	88.08	89.24	50.59	97.92
2	93.84	93.14	93.49	65.88	**99.48**
3	95.54	95.68	95.61	**70.59**	**99.48**
4	**95.77**	**96.05**	**95.91**	**70.59**	**99.48**

(B) PARTIAL PARSES ONLY (7.64%)

	Most Probable Parse (MPP)				
	precision	recall	f-score	exact	tags
1	**70.48**	**72.67**	**71.56**	0	93.55
2	68.67	70.81	69.72	0	92.47
3	69.88	72.05	70.95	0	93.55
4	69.88	72.05	70.95	0	**94.62**
	Most Probable Derivation (MPD)				
	precision	recall	f-score	exact	tags
1	70.48	**72.67**	71.56	0	93.55
2	**70.91**	**72.67**	**71.78**	0	**94.62**
3	70.48	**72.67**	71.56	0	**94.62**
4	69.88	72.05	70.95	0	**94.62**
	Shortest Derivation (SDer)				
	precision	recall	f-score	exact	tags
1	**70.99**	71.43	71.21	0	93.55
2	69.09	70.81	69.94	0	93.55
3	70.48	**72.67**	**71.56**	0	**94.62**
4	70.48	**72.67**	**71.56**	0	**94.62**

Table 3.7: FRENCH. Results in column (A) show the effect on the parse accuracy of sentences which received full parses, for each ranking strategy, of increasing fragment depth. Results in column (B) show the effect on the parse accuracy of sentences which were not assigned full parses, for each ranking strategy, of increasing fragment depth.

– both in terms of the f-score and exact match metrics – using MPP ranking at depth 3.

As was the case for English, the results presented thus far have been calculated over both partial and complete parses. On average, 92.36% of the sentences in each test set are assigned full parses according to the training data and 7.64% receive only partial parses. In order to investigate whether the trends observed over all parses also emerge when we distinguish between partial and complete parses, we present separate evaluations of the quality of complete and partial output in Tables 3.7 and 3.8; the results in column (A) (on the left) refer to evaluation over full parses only and the results in column (B) (to the right) refer to evaluation over partial parses only.

We look first at the trends observed from the results presented in Table 3.5; these show that parse accuracy peaks at depth 3 in terms of precision, recall, f-score and exact match for MPP ranking, that parse accuracy for MPD ranking peaks at depth 4 according to precision, recall and f-score and that accuracy for SDer ranking is highest at depth 4 for precision, recall and f-score and depth 3 for exact match. Column (A) of Table 3.7 shows that exactly the same trends hold when evaluation takes place over full parses only. Conversely, column (B) of Table 3.7, which gives evaluations over partial parses only, is

(A) FULL PARSES ONLY (92.36%)

	F-Score		
	MPP	MPD	SDer
1	92.22	**92.68**	89.24
2	96.10	**96.13**	93.49
3	**97.06**	96.09	95.61
4	**96.65**	96.62	95.91

	Exact Match		
	MPP	MPD	SDer
1	**55.29**	52.94	50.59
2	68.24	**72.94**	65.88
3	**76.47**	70.59	70.59
4	**74.12**	70.59	70.59

(B) PARTIAL PARSES ONLY (7.64%)

	F-Score		
	MPP	MPD	SDer
1	**71.56**	**71.56**	71.21
2	69.72	**71.78**	69.94
3	70.95	**71.56**	**71.56**
4	70.95	70.95	**71.56**

	Exact Match		
	MPP	MPD	SDer
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0

Table 3.8: FRENCH. Results in column (A) show the relative performance of ranking strategies MPP, MPD and SDer as fragment depth increases on sentences which received full parses. Results in column (B) show the relative performance of ranking strategies MPP, MPD and SDer as fragment depth increases on sentences which were not assigned full parses.

much less consistent: MPP f-scores peak at depth 1 rather than depth 3 while MPD f-scores peak at depth 2 rather than depth 4 and SDer f-scores are equally high at depths 3 and 4. (Again, exact match scores for partial parses are always 0 as it is not possible for a partial parse to be identical to its reference parse.)

Trends observed from the results presented in Table 3.6 show that the best f-scores were obtained using MPD at depths 1 and 2 and the MPP at depths 3 and 4 and that the best f-score overall is achieved using MPP ranking at depth 3; these trends are precisely replicated when evaluation is over full parses only, as shown in column (A) of Table 3.8. Conversely, no discernable trends are observed when partial parse f-scores are considered in isolation in column (B) of Table 3.8: exactly the same f-score of 71.56% is achieved using MPP and MPD at depth 1, MPD and SDer at depth 3 and SDer at depth 4 and the best f-score, which is achieved using MPD at depth 2, only improves on this score by 0.22%. Thus, the results over full parses only show similar patterns to those seen over all parses whereas the partial parse results are less predictable. Again, this not an unreasonable balance as full parses are far more likely to be reliable than partial ones.

3.2.5 Discussion and conclusions

In this section, we discuss the parsing results presented above. In particular, we look at how DOP improves over basic PCFG parsing for the same data, we compare our results to previous DOP experiments on the English HomeCentre corpus, we look at the trade-off between accuracy and efficiency, we discuss the DOP Hypothesis and we consider whether DOP is as successful at modeling French data as it is at modeling English. In order to facilitate discussion of these issues, we present the CPU times taken to parse and disambiguate each input sentence for each ranking strategy and fragment depth, the number of samples taken when parsing using random sampling at each fragment depth, the size of the fragment set extracted from each training set at each depth and the CPU times taken to perform this extraction using the method outlined in section 3.1.2.

Does DOP improve over PCFG parsing on the HomeCentre?

Extracting a PCFG from each set of training data and using it to find the most probable PCFG parse for each sentence in the test set corresponds exactly to extracting the set of depth 1 DOP fragments from each set of training data and using the Viterbi algorithm to compute the most probable derivation for each test set sentence. Our English parsing results (given in Table 3.2) show that the best DOP f-score achieved (94.20% at depth 2 using MPP ranking) is 2.76% higher than the corresponding PCFG f-score (91.44%) and the best DOP exact match figure achieved (80% at depth 3 using MPP ranking) is 17.78% higher than the corresponding PCFG exact match score (62.22%). Further increases in parse accuracy are shown for French parsing (given in Table 3.6): the best DOP f-score achieved (94.23% at depth 3 using MPP ranking) is 3.84% higher than the corresponding PCFG f-score (90.39%) and the best DOP exact match figure achieved (72.22% at depth 3 using MPP ranking) is 22.22% higher than the corresponding PCFG exact match score (50%). Thus, we conclude that the DOP model performs better on the English and French HomeCentre corpora than the basic PCFG model.

Do we improve on previous DOP HomeCentre results?

Results for previous DOP experiments on the English section of the HomeCentre corpus were published by Bod and Kaplan (2003). (No parsing experiments on the French section have been published to date.) These experiments were run in order to compare Tree-DOP performance against LFG-DOP performance and, consequently, evaluation of the Tree-DOP results was limited. The experiments carried out used all fragments up to and including depth 4 and the MPP was selected by random sampling; no other ranking strategies were used. The output parses were evaluated by averaging exact match, precision and recall scores over 10 training/test splits; an f-score of 92.75% and an exact match score of 49% were reported. Our f-score and exact match figures for the equivalent experiment (i.e. depth ≤ 4, MPP ranking, 8 training/test splits) are 92.17% and 68.89% respectively. Thus, our f-score is 0.58% lower than that of Bod and Kaplan (2003) but our exact match score is 19.89% higher. Bod and Kaplan (2003) do not report whether the scores at depth 4 were the best scores achieved or whether they achieved higher scores at lower fragment depths; at depth 2 we outperform their f-score by 1.5% and at depth 3 we outperform their exact match score by 31%. Furthermore, their exact match score is 13.22% lower than the lowest exact match score we report (62.22% at depth 1 using MPD ranking – in other words, the basic PCFG model).

There are a number of possible explanations for the differences in results achieved. Firstly, as stated in section 3.2.1, we remove unary productions of the form $X \longrightarrow Y$ by simply replacing the production with the more specific category Y; we do not know if the experiments described in (Bod and Kaplan, 2003) did the same. Secondly, the methods used to control the size of the sample set when approximating the search for the most probable parse are not the same; Bod and Kaplan (2003) use re-scored sampling probabilities and compute at intervals of 100 samples the probability of error such that they are 95% certain that the most frequently sampled parse is the most probable. We, on the other hand, use an exact sampling method which computes the probability of error after each sample is taken until we are 99% certain that the most frequent parse in the sample is the most probable. The differences between these methods are discussed in detail in section 2.5.1. Finally, the experiments reported in (Bod and Kaplan, 2003)

use some simple online pruning (Bod, personal communication) whereby each fragment at each chart position is assigned a score equal to the product of its prior and inside probabilities (i.e. the fragment probability by the total probability mass available at each of its substitution sites) and fragments with scores less than 10^{-5} times that of the best fragment at that chart position are discarded (Bod, 2001). The experiments we report do not employ this pruning technique.

Ranking algorithms: efficiency vs. accuracy

Calculating parse probabilities for trees generated by DOP grammars by summing over all derivations which yield each parse is, in theory, an attractive proposition because it makes use of all the probabilistic evidence present in the fragment base used for training: every fragment which can be used in deriving a given parse contributes to the overall probability of that parse. In practice, however, this proposition is generally considered to be rather less attractive because (i) explicitly calculating the probability of every possible parse to find out which one is the most probable is NP-complete (Sima'an, 1995b) and (ii) using approximations such as Monte Carlo sampling is seen as being less efficient than simply calculating, for example, the most probable derivation using the Viterbi algorithm. Thus, there exists a trade-off between accuracy and efficiency: we expect greater accuracy if we find the MPP but we expect to be able to find the MPD more quickly.

Our results – for both English and French – confirm that greater parse accuracy is achieved by searching for the most probable parse rather than for the most probable derivation but scores using MPD ranking are very close to those using MPP ranking. We see in Table 3.2 that, for English, the highest f-score achieved by finding the MPD (93.67% at depth 2) is only 0.53% lower than the highest f-score achieved by finding the MPP (94.2% at depth 2) and the highest exact match score achieved using MPD (77.78% at depth 2) is 2.22% lower than the highest exact match score achieved using MPP (80% at depth 3). Similarly, we see in Table 3.6 that, for French, the highest f-score achieved by finding the MPD (93.83% at depth 4) is only 0.4% lower than the highest f-score achieved by finding the MPP (94.23% at depth 3) and the highest exact match score achieved using MPD (68.89% at depth 2) is 3.33% lower than the highest exact match score achieved

	ENGLISH				FRENCH		
	CPU seconds/sentence				CPU seconds/sentence		
	MPP	MPD	SDer		MPP	MPD	SDer
1	0.878	0.811	0.8111	1	3.178	3.144	3.022
2	1.811	1.756	1.70	2	4.678	4.611	4.456
3	4.556	4.633	4.40	3	8.167	8.011	7.722
4	17.32	17.44	17.33	4	41.69	41.14	39.38

Table 3.9: Comparison of average sentence processing times (parsing and disambiguation) for each ranking strategy at each depth.

using MPP (72.22% at depth 3).

On the other hand, our experiments do not confirm – either for English or for French – that finding the most probable derivation can be accomplished more quickly than finding the most probable parse. Table 3.9 shows that while calculating the MPP takes slightly longer per sentence, the greatest difference in time taken at each depth is 0.55 seconds for French at depth 4. This somewhat surprising outcome is due mainly to the sampling algorithm implemented – which we have adopted directly from (Chappelier and Rajman, 2003) – whereby the number of samples taken is statistically controlled to within a 1% error rate, i.e. sampling continues until we are 99% certain that the most probable parse according to the DOP model has been found. The average numbers of samples taken per sentence at each depth are given in Table 3.10; for English, average samples taken decrease from just under 30 per sentence at depth 1 to 17.5 at depth 4 and, for French, average samples taken decrease from just over 57 per sentence at depth 1 to 18.2 at depth 4. (This decrease is as expected: the more probabilistic context given to the model, the easier it becomes to discern which parse is most probable.) Clearly, far fewer samples have to be taken than one might have expected – for example, Bod and Kaplan (2003) compute a minimum of 100 samples for every sentence and check the probability of error at intervals 100 samples.

Thus, we conclude that greater accuracy can be achieved when parsing the English and French HomeCentre corpora by ranking output parses according to parse probability and that, furthermore, this can be done without sacrificing efficiency.

ENGLISH

	depth 1	depth 2	depth 3	depth 4
MPP sample sizes:	29.65	18.86	18.91	17.53

FRENCH

	depth 1	depth 2	depth 3	depth 4
MPP sample sizes:	57.23	29.87	22.38	18.23

Table 3.10: Comparison of the average number of samples taken at each depth in order to be 99% sure that the most frequent parse in the set of samples derivations was also the most probable parse according to the DOP model.

What happened to the DOP Hypothesis?

The DOP Hypothesis states that as we increase the size of the set of fragments extracted from the training data by including larger fragments, parse accuracy should also increase. Thus, according to the DOP Hypothesis, as we parse our test sets using increasingly larger fragment sets including fragments of depth 2 or less, 3 or less and 4 or less, we should see corresponding increases in the scores obtained by the output parses. Our results, however, do not confirm this hypothesis. Table 3.1 shows that, for English, all ranking methods exhibit reduced accuracy as fragment depth increases from 3 to 4 and that performance also deteriorates as depth increases from 2 to 3 for MPD ranking and also for MPP ranking for the precision, recall and f-score measures. For French, Table 3.5 shows that peak performance for MPD and SDer in terms of precision, recall and f-score is achieved at depth 4 but for MPD the best exact match score is achieved at depth 3 and for SDer depths 3 and 4 achieve equal exact match scores; MPP performs best at depth 3 for all metrics – this is the best performance overall.

As discussed in section 2.6.1, the vast majority of published DOP experiments confirm the DOP Hypothesis but either used only a subset of the available fragments at each depth or used training sets extracted from smaller corpora which yield (relatively) small increases in the size of the fragment base as fragment size increases. In our experiments, we have only pruned the fragment set by limiting the depths of included fragments which means that we have used the full set of available fragments at each depth. Furthermore, while the HomeCentre corpora are not particularly large in terms of the number of trees they contain, they are large in terms of the numbers of fragments they yield. This is

ENGLISH

	depth 1	depth 2	depth 3	depth 4
Fragments per training set:	13,750	40,958	245,745	4,271,946
CPU seconds to compile fragment set:†	110.9	115.9	126.6	180.0

FRENCH

	depth 1	depth 2	depth 3	depth 4
Fragments per training set:	15,066	42,147	182,832	2,747,277
CPU seconds to compile fragment set:†	112.5	116.6	122.2	170.5

†We have included the information on the time taken to compile each fragment set in order to (i) illustrate the relatively low increase in time required to compile increasingly larger fragment sets and (ii) be as comprehensive as possible in documenting our experiments. However, the absolute values are not informative as this module has not been optimised for speed.

Table 3.11: Comparison of training set details in terms of the number of fragments at each depth and the time taken to compile each fragment set at each depth.

ENGLISH

	%(d=1)	%(d=2)	%(d=3)	%(d=4)
$d \leq 1$	100	-	-	-
$d \leq 2$	33.57	66.43	-	-
$d \leq 3$	5.60	11.07	83.33	-
$d \leq 4$	0.32	0.64	4.79	94.25

FRENCH

	%(d=1)	%(d=2)	%(d=3)	%(d=4)
$d \leq 1$	100	-	-	-
$d \leq 2$	35.75	64.25	-	-
$d \leq 3$	8.24	14.81	76.95	-
$d \leq 4$	0.55	0.99	5.12	93.34

Table 3.12: Comparison of the proportion of the fragment set occupied by each fragment depth (d) as overall fragment depth increases.

illustrated in Table 3.11, where we see that the size of the DOP fragment sets increases dramatically as fragment depth increases, from just 13,750 and 15,066 depth 1 fragments for the English and French corpora respectively to approximately 4.25 million and 2.75 million fragments of depth 4 or less.

In order to better illustrate the impact on the distribution of fragments in the fragment base of this explosion in the numbers of fragments, we provide in Table 3.12 a comparison of the percentage of fragments (in the fragment base) which are of depth d as the overall depth of the included fragments increases. For example, trivially, 100% of the fragments in the fragment base are of depth 1 if depth is restricted to 1, but if the English fragment base is enlarged to also include fragments of depth 2 then those depth 1 fragments account for only 33.57% of the fragments and the other 66.43% of the fragments are of depth 2. As the number of DOP fragments increases exponentially as depth increases, the fragments of maximum depth occupy the largest proportion of the fragment space. Furthermore, as overall maximum depth increases, the proportion of the fragment space occupied by the

fragments of maximum depth also increases and, correspondingly, the proportion occupied by the smaller fragments decreases. This means, for example, that those depth 1 English fragments which comprised 33.57% of the fragment set at $d \leq 2$ occupy 5.60% of the fragment set at $d \leq 3$ and just 0.32% at $d \leq 4$. Correspondingly, while depth 2 fragments at $d \leq 2$ comprise 66.43% of the fragment set, depth 3 fragments at $d \leq 3$ comprise 83.33% and depth 4 fragments at $d \leq 4$ comprise 94.25%.

As discussed in section 2.6, this distribution of fragments in the fragment base impacts on the probability model when the relative frequency estimator is used to estimate fragment probabilities as it also determines the proportion of the probability mass given over to the fragments of various depths. For example, for $d \leq 4$ only 0.32% of the fragment probability mass is allocated to fragments of depth 1, whereas 94.25% is allocated to fragments of depth 4 and, consequently, a bias towards larger parses is introduced into the probability model. As observed in our parsing experiments, the probability model which results does not exhibit the desired behaviour: parse accuracy decreases for all ranking strategies despite the increased contextual information available. Interestingly, we observe from Table 3.10 that the number of samples needed to establish the most probable parse decreases as the size of the fragments in the fragment base increases. In other words, the presence of larger fragments in the fragment base makes it easier to determine which parse is most probable according to the model; it is simply the case that the ranking imposed by the model is increasingly inaccurate as fragment depth increases due to the skewed fragment probability estimates.

Why do MPP and MPD not score the same at depth 1?

When fragments of depth 1 only are included in the fragment base, the set of fragments and relative frequencies correspond exactly to the probabilistic context-free grammar which can be extracted from the training trees. Thus, a depth 1 DOP grammar behaves exactly as a PCFG, in that each unique derivation yields a unique parse tree. This means that the probability of each parse tree is exactly the probability of the unique derivation which yields that parse tree. Consequently, we would expect that, at depth 1, both the MPP and MPD ranking algorithms would select the same best parse. Surprisingly, however, the

results given in Tables 3.2 and 3.6 indicate otherwise: at depth 1 for both English and French, MPD ranking outperforms MPP ranking in terms of f-score whereas the opposite holds, i.e. MPP outperforms MPD, in terms of exact match.

There are two possible explanations for this observation. Firstly, the difference in results may be due to the 1% chance that the most frequent parse in the random sample does not correspond to the most probable parse according to the DOP model. However, it is also possible that the difference in results is due to how the situation where two sub-derivations are equally likely is handled when the Viterbi algorithm is used to determine the most probable derivation. For PCFG parsing, as there is no motivated way to choose between equally likely sub-derivations, these sub-derivations are pruned at random according to the order in which they are processed: a sub-derivation is only replaced if a *more likely* sub-derivation with the same root node is found. We find that, for DOP, this situation arises quite frequently and, consequently, an unquantified random element has been introduced into the selection process for the MPD (and, therefore, the SDer).

Which is harder for DOP? Parsing English or parsing French?

As stated in section 3.2.1, the sentences in the English and French HomeCentre treebanks are translations of each other. Furthermore, the styles in which the treebanks have been annotated are very similar as they were both annotated at Xerox Parc. Accordingly, the differences between the treebanks are language-specific rather than treebank-specific, i.e. the differences are down to the dissimilarities between English and French syntax rather than to differences in text type and/or treebank annotation styles. Having such parallel treebanks for English and for French affords us the opportunity to look at how well DOP models a language other than English in a manner which factors out treebank-specific explanations for performance differences.

The results presented in Tables 3.2 and 3.6 show that the best f-score achieved for English (94.20%) is very similar to the best f-score achieved for French (94.23%). However, exact match scores indicate that the quality of the English parses produced is 7.78% higher than the quality of those produced for French. This may be partially explained by looking at the number of fragments extracted from the English and French treebanks at each depth:

Table 3.11 shows that the French treebank yields far fewer fragments (and, therefore, less contextual information) at depths 3 and 4 than the English treebank. It is also the case, however, that the English treebank sentences are, on average, 1.33 words shorter than the French treebank sentences. This is reflected in the average parse times for each sentence at each depth given in Table 3.9, which show that it generally takes longer to parse the French sentence than the English sentences. Table 3.10 shows that at each fragment depth, fewer samples need to be taken to disambiguate English sentences than French sentences; this indicates that the French sentences are more ambiguous (relative to the training data) than the English sentences. Thus, we conclude that – at least for this corpus type – the DOP model copes better with English text than with French text.

3.3 Summary

In this chapter, we have presented the design of our Tree-DOP parser and documented our motivations for selecting each of the algorithms used. The system we have built is flexible enough to serve as the main technology behind data-oriented models which assume augmented context-free phrase-structure tree representations. We have confirmed its accuracy by ensuring that our English parse scores are comparable to those achieved by Bod and Kaplan (2003) on the same data. We have also performed the first experiments which apply the DOP model to parsing French text[10] and shown that parse scores are similar to those achieved for English, although parse times are generally longer for French than for English. We note that, as the DOP model is language-independent, the same system was used for both the English and French experiments. Furthermore, we observed that the DOP Hypothesis does not hold for parsing experiments on the English and French HomeCentre corpora and concluded that this was a manifestation of the bias in the parameter estimation method employed. Finally, we investigated the trade-off between accuracy and efficiency for DOP and concluded that, for the English and French HomeCentre corpora, greatest accuracy is achieved by outputting the most probable parse and, furthermore, this can be done without sacrificing efficiency.

[10] Although not presented here, we have also performed preliminary experiments on parsing Chinese text with the DOP model; these experiments are documented in (Hearne and Way, 2004).

In the next three chapters, we study the data-oriented model of translation – which was inspired by DOP – in theoretical, practical and performance terms. We discuss how it relates to other approaches to machine translation, how the implementation described here influences our translation system design, and how the translation model performs. In chapter 7, however, we return to the topic of parsing. In this chapter, we study the DOP model which assumes and generates LFG representations rather than context-free phrase-structure trees. The issues raised and conclusions drawn in chapters 2 and 3 regarding the Tree-DOP model will be referred to throughout the rest of this thesis.

Chapter 4

Data-Oriented Translation

In section 4.1 of this chapter, we describe the main machine translation (MT) paradigms – rule-based MT and data-driven MT – in current use, and discuss methods of creating hybrid models which combine elements from both. We then outline the general principles which underlie the Data-Oriented Translation (DOT) model of MT and give a precise description of a particular instantiation of this model, Tree-DOT, in section 4.2. Having presented the Tree-DOT model in detail, we then discuss how it relates to both the rule-based and data-driven MT paradigms, and pin down the similarities and differences between DOT and other models.

4.1 Paradigmatic approaches to MT

There are two main paradigmatic approaches to MT. Broadly speaking, **rule-based** systems translate by following a set of instructions provided by linguistic experts whereas **data-driven** systems learn from example sentences translated by professional translators. In this section, we discuss the strengths and weaknesses of these paradigms and look at hybrid approaches which seek to take the best from each.

4.1.1 Rule-based MT

Rule-Based MT (RBMT) is characterised by the use of rules, generally hand-written by linguists, in order to translate between languages. Translations are produced by analysing the input – levels of analysis vary from very shallow to deep – using rules to translate

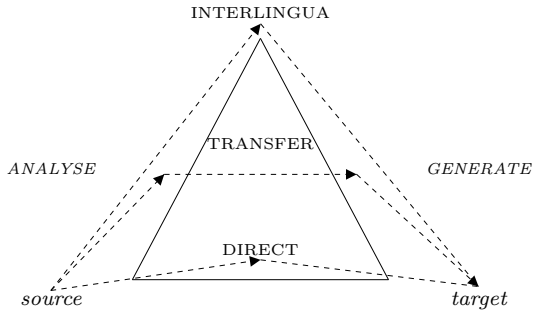


Figure 4.1: The *Vauquois pyramid* (Vauquois, 1968) summarises the relationships between the direct, transfer-based and interlingua-based approaches to Rule-Based MT.

the analysis into a target-language analysis and then generating an output string. Within the RBMT paradigm there exist three main translation strategies: direct MT, transfer-based MT and interlingua-based MT. These methodologies differ quite significantly, both in terms of the types of linguistic information they assume and in how they put this information to use.

The **direct** approach calls for a bilingual dictionary and a small number of rules characterising target-language word order. The source language words are replaced with the translations found for them in the dictionary and then the string is rearranged so that its word order is appropriate for the target language. The **interlingua** approach requires a powerful analysis component which takes the string to be translated and assigns to it an abstract linguistic representation which is independent of both the source and target languages. The generation component then produces the appropriate target-language translation from this representation. The **transfer** approach calls for an analysis component which assigns to the source-language input string an intermediate representation which is reasonably abstract yet not language-independent. A transfer component comprising mapping rules then translates this source-language intermediate representation into a target-language intermediate representation from which the output translation is generated.

As illustrated in Figure 4.1 using the *Vauquois pyramid* (Vauquois, 1968), the main difference between these strategies lies in the extent to which they abstract away from the source-language input strings. The direct method takes the shortest possible route

from source to target strings; little or no source-language analysis is performed and, consequently, target-language generation is trivial. In contrast, the interlingua method takes the longest possible route from the source string to the target string, forming a completely abstract, language-independent representation of the meaning of the input sentence from which a target string can be generated. The transfer method occupies the middle ground between these two extremes; analysis of the source string is deeper than for the direct method but, nevertheless, its representation remains language-dependent, while translation between intermediate representations, which is unnecessary for the interlingua method, requires more sophisticated transfer mappings than the reordering rules used in direct translation.

In practice, the distinction between the various instantiations of these methods is far less clear cut due to the fact that the type of intermediate representation used in a transfer system determines how far it abstracts from the source string and, consequently, how much work has to be done during the analysis, transfer and generation stages. For example, if a reasonably shallow representation is produced, then less work is done during analysis but transfer from source to target representation is likely to require more work. In contrast, a very detailed source-language analysis is likely to make the task of transfer less onerous, but highly abstract target-language representations can prove challenging for generation.

MT systems which use RBMT techniques are capable of producing translations of reasonable quality due to the fine-grained and sophisticated nature of the linguistic rules they employ. This quality, however, comes at a high price: RBMT systems are expensive to build precisely because of the degree of linguistic sophistication they require. Each component must be hand-coded by linguistic experts who have knowledge of either the source language, the target language, or both. Furthermore, these components are often useful only for the language pair, language direction and text type for which they were initially developed; switching to other languages and genres can often mean starting from scratch. Extending hand-coded components to widen coverage can also be problematic as it is frequently not possible to predict how newly-added rules will interact with those already in use. Widening coverage is, however, crucial to the success of RMBT systems because they tend not to be robust: if the input is either ill-formed or simply beyond

the scope of the rules then the system will fail to generate a translation. This lack of robustness is also an issue in the opposite situation where more than one translation can be generated for the input string, as there is often no means of indicating which translation is the 'best' translation.

4.1.2 Data-driven Machine Translation

Within the data-driven MT paradigm there exist two main translation strategies: statistical MT (SMT), and example-based MT (EBMT). These methodologies are grouped together because, in contrast to the rule-based approaches, they generate translations for input strings using evidence gathered from monolingual and bilingual collections of text. Again, although similar in that they both acquire translation knowledge from previously-occurring utterances and their translations, these methods differ significantly, both in the type of information learned and how this information is used.

Statistical MT systems invoke models of both language and translation from large quantities of (monolingual and bilingual parallel) data using highly-developed theories of probability distribution and estimation. The translation model is used to establish the set of target-language words which are likely to be useful in translating the input string and the language model is used to select for output the string that is most likely to be generated from this set of target-language words. Training the language model involves establishing the frequency distributions of all n-grams (i.e. word sequences of length n) occurring in the monolingual training data; bigram or trigram models are usually used. The translation model, on the other hand, tends to be more sophisticated, taking into account features such as source and target word co-occurrence frequencies, sentence lengths and the sentence positions in which words occur. The translation model is usually trained on bilingual data comprising raw (i.e. unannotated) strings aligned at sentence level.

Example-based MT systems also assume large quantities of bilingual data aligned at sentence level – usually referred to as an example base – but translate by the principle of analogy: translation is performed by adapting past translations of similar input. Trivially, each input string is matched against the source side of the corpus and if the sentence is found then its corresponding translation is output. However, things are not usually

this simple. If an exact match is not found then sentences which are similar to the input string are identified, their translations retrieved and phrases or chunks from those translations combined to form an output translation. When comparing the source string to the source side of the example base, measures of similarity are usually based on such clues as word co-occurrences, part-of-speech tags and correspondence to generalised example templates. Recombining the example translations involves identifying those segments of the example target string which correspond to the matched segments of the example source strings, meaning that sub-sentential alignments are necessary. These can be identified using heuristics and/or resources such as dictionaries and thesauri either off-line or during translation. Furthermore, empirically-established weights can be assigned to translation chunks and if there are multiple translations for an input string then these translations can be ranked according to their weights.

Of course, SMT and EBMT techniques can be combined. For example, the EBMT matching process – which generally requires less training data than the corresponding SMT translation model – can be used to retrieve translations of sentences similar to the input string, and then a statistical language model (trained on monolingual data) used to generate the best possible output string. EBMT techniques have also had an impact on SMT, resulting in translation models which work at the phrase level (corresponding to EBMT chunks) rather than solely at word level.

As data-driven methods of translation are fundamentally string-based, they lack the linguistic sophistication which allows RBMT systems to output high-quality translations. As target-language strings are constructed either using word-lists coupled with statistical information regarding the likelihood of their possible combinations and orderings (SMT) or using words and phrases which follow the order of the source-language words and phrases they are translations of (EBMT), the output translation may be ill-formed. Generally, the further apart dependencies are in the output string, the less likely these methods are to output well-formed translations. However, data-driven systems are easier and cheaper to build than RBMT systems because, while good-quality bilingual data in sufficiently large quantities to be useful for training models and/or as an example base is not easy to come by, hand-coding RBMT system components is much more difficult. Furthermore, data-

driven systems can be extended by adding more data, or used for different language pairs and text types by replacing the data with appropriate material and, in the case of SMT, re-training the models. (In theory, no changes need be made to the translation algorithms themselves.) Another important issue is that of robustness: unlike rule-based systems, data-driven systems will generally produce the best translation possible no matter what the input, giving confidence scores to alternative translations according to their weights or probabilities.

4.1.3 Hybridity: the best of both worlds

The data-driven methods (i.e. SMT and EBMT) are robust: they will always produce some translation no matter what the input string. This makes such systems very attractive; if an RBMT system does not find a sequence of rules which can be applied successfully to the input then no translation will be produced. Another attractive characteristic of the data-driven methods is ease of knowledge acquisition. RBMT systems are time-consuming and expensive to build and difficult to maintain and update, whereas it is much easier to acquire raw data. However, statistical and example-based systems are not good at modeling linguistic phenomena such as agreement, even at short distances. RBMT systems, in contrast, can handle linguistic phenomena such as agreement, even at longer distances. Thus, the merits of combining the positive elements of the rule-based and data-driven approaches to MT are clear: a combined model has the potential to be highly accurate, robust, cost-effective to build and adaptable. While the merits are clear, however, how best to combine these techniques into a model which retains the positive characteristics of each approach while inheriting as few of the disadvantages as possible remains an open question.

The analysis, transfer and generation rules traditionally hand-crafted for RBMT can be (semi-)automatically induced from corpora. For example, a set of hand-crafted transfer mappings can be coupled with an analysis component comprising a probabilistic grammar extracted from a treebank. Alternatively, the transfer mappings themselves can also be extracted automatically, this time from bilingual aligned data. Clearly, while such techniques use data to acquire knowledge, thereby speeding up development time and reducing

costs, the translation engines themselves are still rule-based.

There are many possibilities for integrating rule-based techniques into example-based MT systems. For example, EBMT systems can be modified to incorporate target-language grammars – built either automatically or by hand – into the recombination component (e.g. (Bond and Shirai, 2003)). This improves the quality of the output strings by ensuring that the output translation is not only the most likely but also grammatically correct. Alternatively, linguistic information can be imported directly by using aligned annotated text rather than simply aligned sentences. Such systems can, for example, store aligned pairs of word-dependency structures (e.g. (Sato, 1995; Menezes and Richardson, 2003)). In this situation, a parser is used to analyse the input string and the parser output matched against the source representations in the example base. The retrieved target dependency structures are then combined to produce the output translation. In fact, in their discussion on the classification of EBMT systems, Turcato and Popowich (2003) point out that the majority of EBMT systems make use of other resources besides an aligned example base; such resources include bilingual lexica, thesauri, morphological analysers, taggers and syntactic parsers. Accordingly, very few EBMT systems work solely on the level of aligned sentence pairs. Although incorporating many resources which are also used in RBMT, these models are still classed as example-based models because they also use knowledge from the example base such that this knowledge cannot be determined before the input string has been seen.

Statistical MT systems, on the other hand, have traditionally only made use of the statistics gathered from the data and neither imported linguistic resources nor generated further linguistic resources from the data. This situation has, however, changed slightly in order to incorporate information about the structure of language into the models (e.g. (Yamada and Knight, 2001; Charniak et al., 2003; Melamed, 2004)). The translation model of Yamada and Knight (2001) assumes bilingual aligned sentence pairs where each source sentence has been syntactically parsed. The model transforms a source-language parse tree (i.e. an input string which has been parsed in a pre-processing step) into a target language string and the best translation is determined by the language model. In (Charniak et al., 2003), a syntax-driven language model which generates the best target

string is employed in conjunction with this syntax-driven translation model.

4.2 Data-Oriented Translation: relating linguistics, statistics and examples

The Data-Oriented Translation (DOT) model (Poutsma, 1998, 2000, 2003), instantiated as the Tree-DOT model in section 4.2.1, combines examples, linguistic information and a statistical translation model and, thus, can be described as a hybrid model. The motivations for adopting this model are precisely as before: it combines the robustness of data-driven methods, the experience-based philosophy of EBMT, the probabilistic models of SMT and the linguistic information (to varying degrees) which lends RBMT systems their accuracy. It does, however, differ from other approaches in that it is not allied to any one of RBMT, EBMT and SMT over the others, but rather inextricably interweaves the philosophies of all three in an integrated framework. In short, in the DOT model, none of the three elements – linguistics, statistics and examples – plays a more or less important role than the others.

4.2.1 The Tree-DOT model

In this section, we present the model of MT called Tree-DOT which is based on the Tree-DOP model of parsing. This model was originally described in (Poutsma, 1998) and further details and refinements were given in (Poutsma, 2000, 2003); the description given here is based on (Poutsma, 2003).

As for DOP, providing a specification of the DOT model means we must specify four elements: the type of representation we expect to find in the example base, how fragments are to be extracted from those representations, how extracted fragments are to be recombined when analysing and translating new input strings, and how the resulting translations are to be ranked. In the following, we provide details of each of these elements for the Tree-DOT model.

Representations

As for DOP, many different linguistic formalisms can be used to annotate the example base which underpins any DOT system. Here, as before, we assume context-free phrase structure tree representations. However, representations for the DOT model comprise *pairs* of trees rather than the single trees used for DOP, i.e. we assume a bilingual aligned treebank such that each tree pair represents an example translation pair. Where the languages in our bilingual treebank are L_1 and L_2, all of the trees on the left of our tree pairs represent L_1 strings and all of the trees on the right represent L_2 strings. Links between nodes in L_1 trees and L_2 trees denote translational equivalences: node A_x in an L_1 tree and node B_y in the corresponding L_2 tree are linked if the substrings they dominate can be considered translations of each other. In other words, we assume that the tree pairs are aligned not only at sentence level but also at sub-structural level. While the links between tree pairs are non-directional – i.e. these linked tree pairs can be used when translating either from L_1 to L_2 or from L_2 to L_1 – we generally refer to the L_1 representations on the left of the bilingual treebank as *source* representations and to the L_2 representations on the right as *target* representations.

An example DOT representation comprising a pair of linked trees is given in Figure 4.2. Here, as the left tree represents an English string, we refer to English as the source language and, correspondingly, French as the target language. As this example illustrates, not every node is (nor should be) linked to a node in the corresponding tree. Frequently, there is no node which dominates exactly the translational equivalent of a particular substring. For example, the source substring *press and release*, dominated by the node V, is translationally equivalent to the target substring *exercez une pression brève sur* but there is no node in the French tree which exactly spans this substring. It is also frequently the case that a particular substring simply does not have an overt realisation in the corresponding sentence. For example, we see in Figure 4.2 that the substring *de* dominated by node P in the target tree has no corresponding substring in the source tree. Thus, although we stipulate that there must be links between the root nodes of each tree pair, a minimally-linked tree pair will be linked *only* at sentence level. Note, also, that no links connect terminal symbols directly; this means that words are never separated from

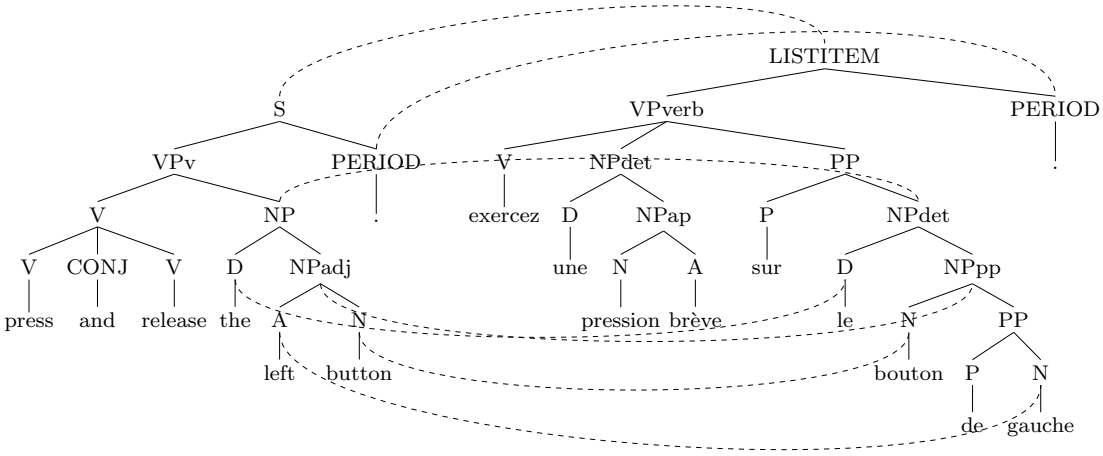


Figure 4.2: Each DOT representation comprises linked source- and target-language phrase-structure trees where the links between source and target nodes indicate that the substrings dominated by these nodes are translationally equivalent.

their part-of-speech tags. The importance of these features of the linking between trees will become clearer as we show how fragments are extracted and combined.

Fragmentation

The fragmentation process involves extracting pairs of linked generalised subtrees from the linked tree pairs contained in the example base. A fragment $<t_s,t_t>$ extracted from tree pair $<T_s,T_t>$ is valid only if it meets the following criteria:

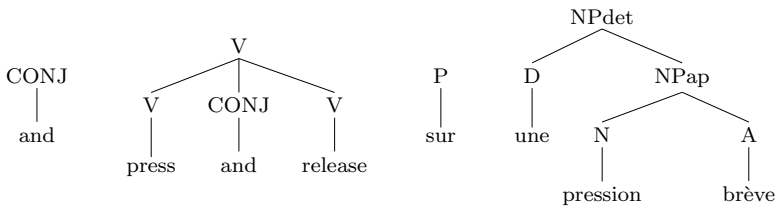
1. each node in t_s is a node in T_s and each node in t_t is a node in T_t,
2. each **linked** node in t_s either has no children or has exactly the same number of children as the corresponding node in T_s and each **linked** node in t_t either has no children or has exactly the same number of children as the corresponding node in T_t,
3. each **unlinked** node in t_s has exactly the same number of children as the corresponding node in T_s and each **unlinked** node in t_t has exactly the same number of children as the corresponding node in T_t, and
4. both t_s and t_t have more than one node.

The difference between criteria 2 and 3 is crucial: linked nodes are not required to have children whereas unlinked nodes must always have exactly the same set of children as in

the original treebank representation. Clearly, then, we must differentiate between linked nodes and unlinked nodes when extracting fragments. The *root* and *frontier* operations defined for Tree-DOP have to be adapted to reflect this difference because, for Tree-DOP, all nodes are treated equally. These modified operations are defined as follows:

- given a copy of tree pair $<S,T>$ called $<S_{copy},T_{copy}>$, select a **linked** node pair $<S_N,T_N>$ in $<S_{copy},T_{copy}>$ to be *root* nodes and delete all except these nodes, the subtrees they dominate and the links between them, and
- select a set of **linked** node pairs in S_{copy} to be *frontier* nodes and delete the subtrees they dominate.

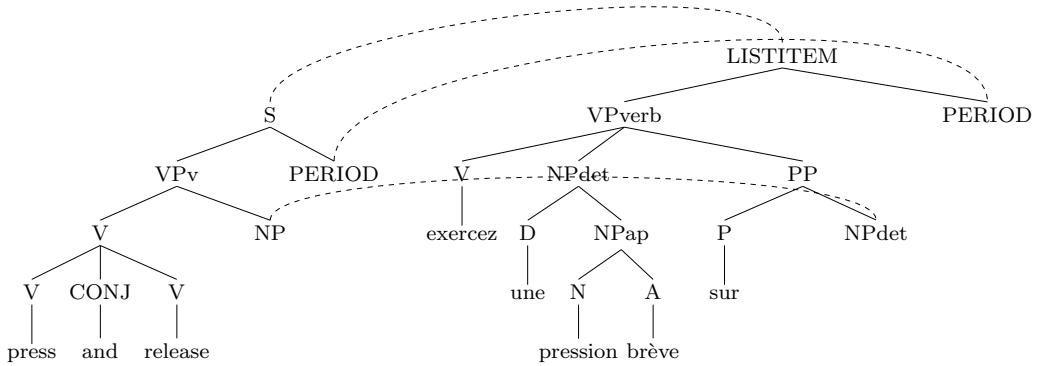
As these operations can be applied to linked node pairs only, well-formedness criterion 3 is never violated because the only way the children of an unlinked node could be deleted is if that node was selected to be a frontier node. Consequently, every fragment $<f_s,f_t>$ extracted comprises two subtrees that are exactly translationally equivalent, i.e. the root nodes of f_s and f_t are linked, every non-terminal frontier node in f_s is linked to exactly one non-terminal frontier node in f_t and every non-terminal frontier node in f_t is linked to exactly one non-terminal frontier node in f_s. This effectively means that subtrees from the representation in Figure 4.2 such as those given in (4.1) – which are valid monolingual fragments according to the DOP model – do not exist in the DOT fragment base because they do not have corresponding subtrees to which they are translationally equivalent. (This reflects the fact that, for DOT, we are interested in expressing translational dependencies rather than monolingual dependencies.)



(4.1)

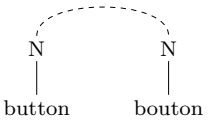
Any lexical item can only be translated in contexts which have been seen before. For example, if the word *release* only appears once in our bilingual treebank, in the representation in Figure 4.2, then the least specific context in which this word appears is the fragment given in example (4.2) below. If this is the case then we can only translate the

word *release* when it appears in the context of the conjoined phrase *press and release* and is followed by a noun phrase.



(4.2)

As source and target terminal symbols are not directly linked in DOT fragments, the minimum context specified about any source-target word pair is their part-of-speech tags. For example, the fragment in (4.3) – also extracted from the representation in Figure 4.2 – gives a direct word-for-word translation pair representing the English word *button* and the French word *bouton*. However, this fragment can only be used where the (English) source word *button* is a noun; if the context in which this word appears signals that it is a verb (as in, *I just need to button my coat*) then this fragment cannot be used when building a translation.



(4.3)

An example of a bilingual aligned treebank is given in Figure 4.3(A) and the full set of DOT fragments which can be extracted from it is given in Figure 4.3(B).

Composition

Following the above description of how DOT fragments are extracted, each unlinked fragment frontier node must be a terminal symbol and each linked fragment frontier node must be a syntactic category. In composition terms, each pair of linked frontier nodes constitutes an open substitution site; fragments whose linked source and target root nodes are of the same syntactic category as the linked source and target substitution site categories can be substituted at these frontiers. For example, fragment f_8 in Figure 4.3(B) has one open substitution site at the linked frontier node pair of category $<N,N>$; any fragment

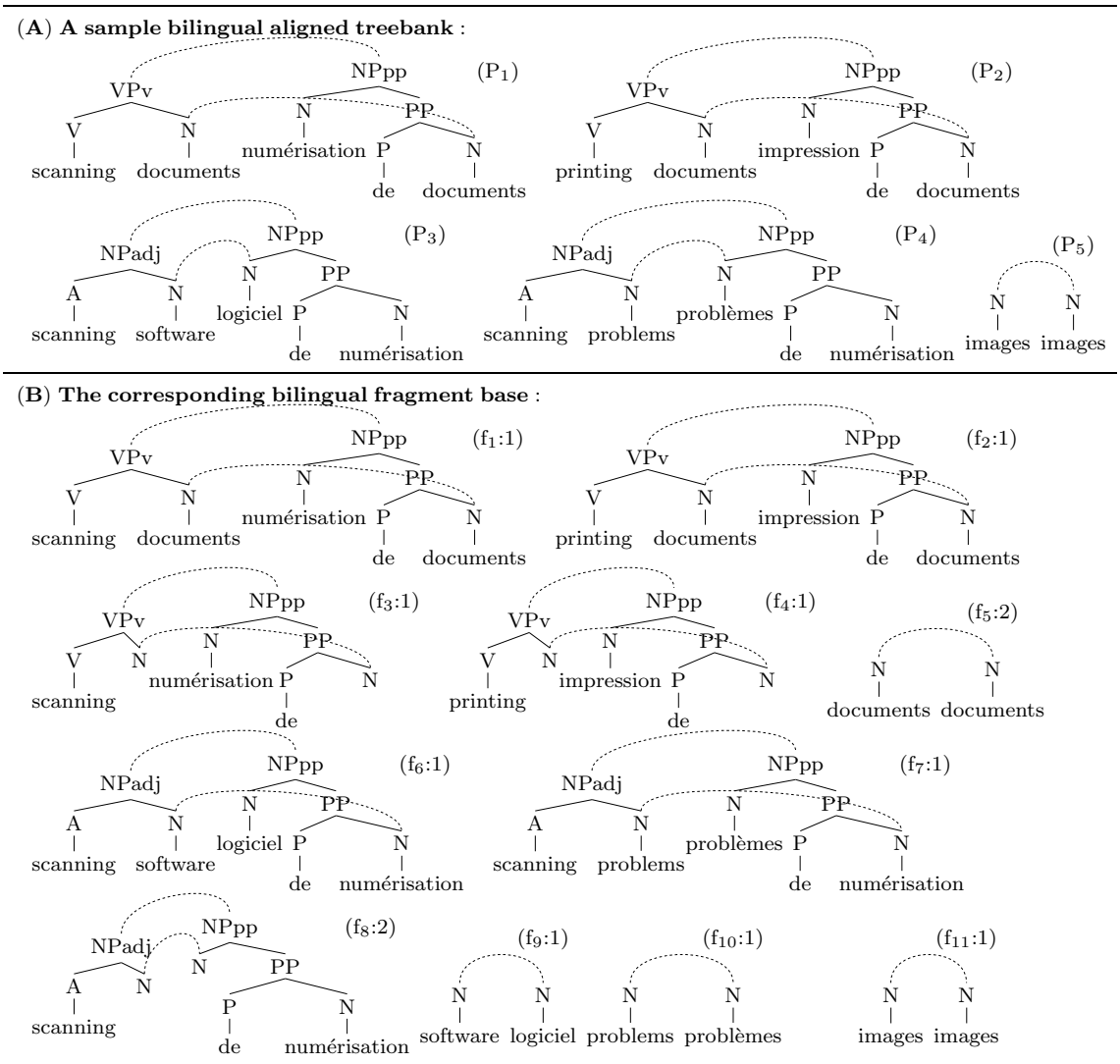
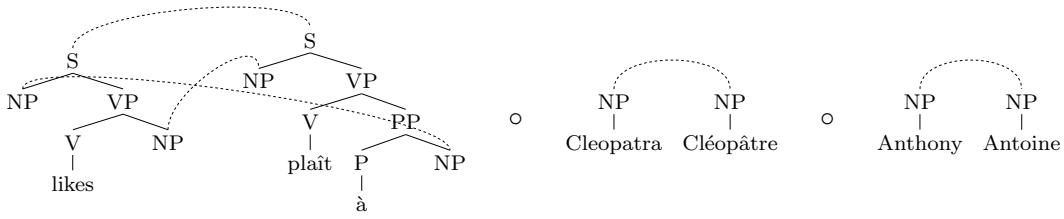


Figure 4.3: Illustration of the process of representation, fragmentation, composition and ranking for Tree-DOT.

in the fragment base whose source and target root nodes are both N can be substituted at this site simply by replacing the source N with the source fragment and the target N with the target fragment.

The Tree-DOP composition operation (∘) is similar to the one defined for Tree-DOP but, again, is adapted to handle linked tree pairs. It is a leftmost substitution operation in that where a fragment has more than one open substitution site, composition must take place at the leftmost site on the source subtree of the fragment. Furthermore, the synchronous target substitution must take place at the site *linked to* the leftmost open source substitution site. This ensures both that each derivation is unique and that each translation built adheres to the translational equivalences encoded in the example base. For example, if the composition operation did not specify order then the composition sequence given in example (4.4)[1] would have two realisations, one where *Cleopatra* is in subject position and the other where *Cleopatra* is in object position (Way, 1999).



(4.4)

Furthermore, this example clearly illustrates the importance of performing target side substitution at the target node which links to the leftmost source substitution site rather than at the leftmost target substitution site. In this example, we see that the subject in the source tree *Cleopatra* translates as the prepositional object *Cléopâtre* in the target tree and the source object as the target subject. If we were to also perform target substitution at the leftmost target site then an incorrect translation would result, with subject translated as subject and object as an oblique object.

Tree-DOT derivations are built by simultaneously building source and target representations using the composition operation. Once an initial fragment is chosen to start the derivation, further fragments are successively substituted at the leftmost source open substitution site and its linked target counterpart until no open substitution sites remain. The

[1]Thanks to Harry Somers for his 'Antony and Cleopatra' example!

output translation associated with each derivation is extracted by simply concatenating the frontier nodes of the target tree. Example derivations are given in Figure 4.3(C).

The probability model

Frequently, multiple representations (i.e. parses and translations) are assigned to an input string. For example, in Figure 4.3(C) we see that two distinct representations are generated for the string *scanning images* as the phrase can be analysed both as a noun phrase and as a verb phrase and the translations differ for each analysis. The probabilities associated with each representation are calculated and used to rank the set of translations and discern which is the most likely translation of the input string.

Firstly, the probability of a fragment is its relative frequency in the set of fragments as given in equation (4.5).[2]

$$P(< s_x, t_x >) = \frac{| < s_x, t_x > |}{\sum_{root(s)=root(s_x) \wedge root(t)=root(t_x)} | < s, t > |} \tag{4.5}$$

The probability of each derivation is then defined as the product of the probabilities of the fragments used to build that derivation as given in equation (4.6).

$$P(D_x) = \prod_{<s_x,t_x> \in D_x} P(< s_x, t_x >) \tag{4.6}$$

The probability of a representation (i.e. a pair of source and target trees) is the sum of the probabilities of the derivations which yield that representation as given in equation (4.7).

$$P(< S_x, T_x >) = \sum_{D_x \ yields \ <S_x,T_x>} P(D_x) \tag{4.7}$$

Finally, the probability that source string s translates as target string t is the sum of the probabilities of the representations which yield both s and t, as given in equation (4.8).

$$P(s,t) = \sum_{<S_x,T_x> \ yields \ s,t} P(< S_x, T_x >) \tag{4.8}$$

[2] As discussed for Tree-DOP in chapters 2 and 3, estimating fragment probabilities according to their relative frequencies in the fragment base is not desirable. In chapter 6, we discuss the ramifications of this estimation method for Tree-DOT.

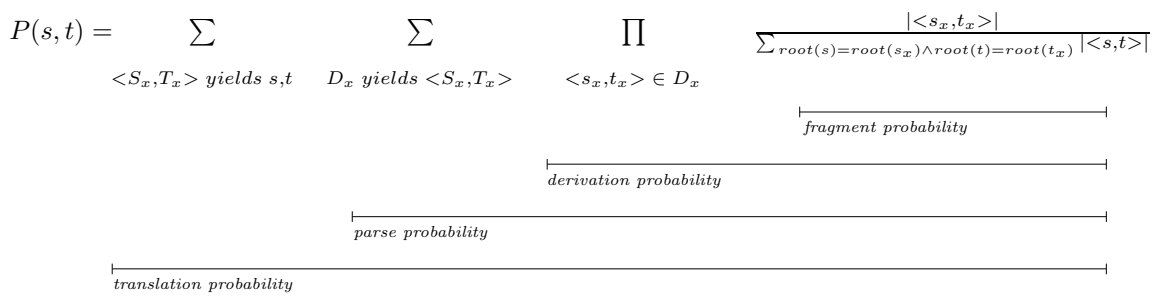


Figure 4.4: The Tree-DOT probability model.

The Tree-DOT probability model is summarised in Figure 4.4.

The calculation of derivation probabilities is illustrated in Figure 4.3(D). Note that in this example each representation and translation has only one derivation, meaning that in this case, derivation probability, parse probability and translation probability are all equal. The second analysis and translation, where the input string is interpreted as a noun phrase, is twice as likely as the first, where the input string is interpreted as a verb phrase. These differing analyses yield translations which have completely different meanings, the second of which is also twice as likely as the first.

4.2.2 DOT: a holistic approach to hybrid MT

At the start of section 4.2, we introduced DOT as a hybrid model of translation which combines elements of RBMT, EBMT and SMT into an integrated framework. Having presented an instantiation of the DOT model, we now discuss in turn how DOT relates to each of these approaches, and highlight how the model is driven by all three equally.

DOT as a transfer-based model of translation

Tree-DOT fragments, which provide snapshots of the translation relationships present in a bilingual aligned treebank, correspond to structural transfer rules. Some of these snapshots, particularly many of those which are not lexicalised, capture very general translation dependencies whereas others – such as the fragment in example (4.2) – are highly specific. In fact, as for DOT, the 'transfer rules' of a Synchronous Lexicalised Tree-Adjoining Grammar (Abeillé et al., 1990) comprise linked syntactic subtrees reflecting syntactic and

functional dependencies not easily captured using localised rewrite rules. However, unlike hand-coded transfer components, DOT shares the DOP philosophy of taking all translational dependencies to be found in the training data into account, rather than just those translational dependencies of perceived importance.

Furthermore, DOT fragments encapsulate differences in word order between the source and target languages, some of which are due to syntactic differences between the languages and some to stylistic differences across text types. While transfer-based systems usually reflect syntactic differences with a good degree of success, stylistic differences are a different matter. It is generally the case that such systems adhere quite closely to the basic structures of the source language when formulating a target language translation (Hutchins and Somers, 1992). DOT, on the other hand, formulates translations on the basis of the evidence in the fragment base, meaning that it will model stylistic as well as syntactic differences according to the evidence presented.

The transfer module in a rule-based system is generally invoked only after the input string has been parsed; the transfer rules are then used to convert from the source parse to a target parse. If we wish to consider DOT fragments as complex transfer rules then, effectively, we form our source-language analysis using exactly the same grammar as we use to translate it. Thus, the analysis and transfer processes are collapsed into one. This means that we never assign to the input string an analysis which cannot be translated to a target language analysis during transfer. This would appear to be more efficient than analysing and transferring separately, as only one grammar is invoked and failing analyses are never generated. Furthermore, as the target subtrees effectively impose constraints on how the source subtrees are combined, these target subtrees actively help in disambiguating the source string.

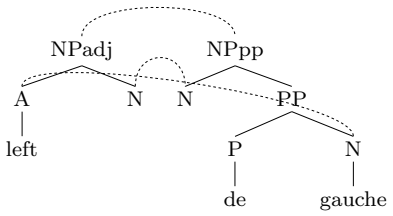
With regard to the handling of input that is not recognised by the model, we note that the DOT model has, in common with rule-based approaches in general, the capacity to signal that the input string is ill-formed. (This is discussed further in section 6.1.) However, the DOT model can also be adapted in order to generate the best translation possible for input which is ungrammatical according to the data.

Finally, it is rarely the case that transfer-based MT systems work solely on the level

of syntactic structure. While the instantiation of the DOT model presented here does exactly that, DOT models can also be defined for representations corresponding to more sophisticated linguistic formalisms – we discuss one such model, which assumes the representations of Lexical-Functional Grammar, in chapter 8. Thus, while the Tree-DOT model is limited in terms of linguistic description to context-free phrase-structure trees, we note that the data-oriented approach to translation in general is bound by no such limitations.

DOT as an example-based model of translation

The aligned bilingual data assumed by the Tree-DOT model also corresponds to the sentence-aligned example base assumed by EBMT models of translation, although EBMT models do not necessarily assume parsed examples. Furthermore, there is a great deal of similarity between the generalisations derived from the data by each method. For example, fragments which also occur as treebank tree pairs correspond to EBMT sententially-aligned examples, fragments which contain no open substitution sites (i.e. all frontiers are terminals) but do not span full sentences correspond to EBMT phrasal alignments and fragments containing open substitution sites correspond to EBMT generalised templates.[3] We illustrate in example (4.9) how a DOT fragment can be seen to correspond to a generalised template (e.g. (Brown, 2003; Way and Gough, 2003)): the (linked) frontiers of the DOT fragment on the left – extracted from the representation in Figure 4.2 – match the template on the right.



left N $\leftrightarrow$ N de gauche (4.9)

There are two main stages in the EBMT translation process. The first – matching – retrieves those examples in the example base which are similar to the input string and the second – recombination – combines the translations of those similar examples into a target-

[3] One of the most significant differences between the DOP model and other experience-based parsing models is that DOP fragments capture dependencies between arbitrary numbers of words. We note that, while DOT fragments also capture dependencies between arbitrary numbers of (source and target) words, this is not novel for MT as EBMT systems do likewise.

language translation. These stages roughly correspond to the DOT tasks of establishing which fragments can participate in deriving (bilingual) representations of the input string and then selecting the best target-language string to output. However, the EMBT and DOT models tackle these tasks in significantly different ways.

The EBMT matching process is generally accomplished using similarity measures based on such clues as word co-occurrences, part-of-speech tags and correspondence to generalised examples. Even when an EBMT model assumes parsed examples, the input string is parsed in a pre-processing step and then the resulting analysis matched against the example base. In contrast, and as previously stated, the DOT model uses the fragments in the fragment base to parse the input string directly. Furthermore, there is no DOT task corresponding to the EBMT task of establishing the required translation segments of each partially-matched example as DOT fragments comprise both source and target subtrees.

The recombination process in the EBMT model and the corresponding composition and disambiguation processes in the DOT model are also driven by quite different strategies. As discussed above, both methods derive units of translation information which vary with regard to size and degree of specificity from the bilingual data they assume. However, they do not prioritise the use of this information in the same way. Generally, EBMT models prefer to recombine as few examples as possible when forming a translation, i.e. those retrieved examples which match the longest word sequences in the input string are chosen for use in recombination. Furthermore, once the smallest set of examples spanning the input string has been found, no more examples are retrieved. Thus, trivially, if the input string occurs exactly in the example base then its translation is retrieved and output without doing any further work. Example weights generally only play a part in selecting the output translation if two or more sets of examples of the same size span the input string. In contrast, the DOT model searches for the most probable translation by summing over the probabilities of all possible translation derivations, irrespective of the number of fragments required to build each derivation. Thus, the DOT model works as hard to translate a sentence spanned by exactly one fragment as it does to translate sentences where long derivations are required.

DOT as a statistical model of translation

The SMT translation model, trained on bilingual sententially-aligned data, is used to establish target-language correspondences for each word in the input string. The SMT language model is then used to select the most likely string from this set of target-language words. The DOT model, on the other hand, makes no probabilistic distinction between translation and generation as each composition sequence explicitly orders the words in the output translation. However, DOT and SMT are similar in that, unlike many EBMT systems, they both comprise full probability models. In addition, they both search for the most probable translation (DOT in one step and SMT in two steps) and, thus, both work hard to translate sentences occurring exactly in the training data.

More recent SMT models which incorporate information about the structure of language into the language and/or translation models are even more similar to the DOT model. Yamada and Knight (2001) use structural information about the source language only (transforming the parsed input into a target string in the translation model), whereas Charniak et al. (2003) also assume structural information in the language model. However, these syntactic models of SMT are similar to EBMT models which assume parsed example bases, in that the input string is parsed as a pre-processing step rather than as an integral part of the translation process. Furthermore, source structural knowledge is applied independently of target structural knowledge; this contrasts with the integrated manner in which the DOT model applies this information.[4]

4.3 Summary

As discussed, the DOT model has much in common with the transfer-based, example-based and statistical approaches to MT. However, DOT also differs significantly from each of these models. Uniquely, the DOT model constitutes a holistic methodology for hybrid MT, in that it depends equally on linguistic information, examples of previously-seen translations and a statistical model of translation. Consequently, this model shares some

[4]The DOT model also bears similarities towards the bilingual stochastic inversion transduction grammars of Wu (1997). However, Wu (*op. cit*) places these in the context of parsing parallel corpora and states that they remain inadequate as fully-fledged translation models.

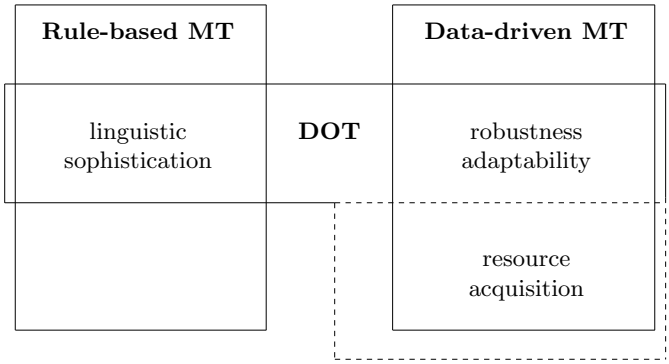


Figure 4.5: DOT as a hybrid model of MT.

of the characteristics of rule-based MT and some of the characteristics of data-driven MT. As summarised in Figure 4.5, the DOT model has the capacity to combine the linguistic sophistication of rule-based models of translation with the robustness and adaptability of data-driven methods. In particular, DOT is a language-independent model, meaning that no adaptation is required when changing to a different language pair, language direction or text type.[5] Furthermore, as for SMT, the model assumes no resources beyond what can be learned from the data provided. DOT does, however, require annotated examples, meaning that resource acquisition is more onerous than for SMT and some models of EBMT. While linguistically-annotated bitexts are not currently freely available, we believe that automatic knowledge acquisition for DOT is viable, and we address this issue further in section 6.6.

In chapter 5, we discuss previous evaluation of the DOT model, assess the reasons behind its disappointing performance and describe an implementation of this model which allows for a more intensive evaluation. In chapter 6, we present experiments comprising a greater degree of translational complexity than heretofore and discuss in detail the results achieved.

[5] In fact, the DOT model has been used to generate paraphrases for English strings by using English as both the source and target languages (Finch et al., 2003).

Chapter 5

Tree-DOT in practice

The issues which arise when implementing a Tree-DOP system – efficiently extracting fragments from a treebank, computing the DOP parse-space for an input string and selecting the best parse for the input according to the DOP model, as discussed in chapters 2 and 3 – also arise when implementing the Tree-DOT model. In section 5.1 of this chapter, we outline the system developed by Poutsma (2000, 2003) in terms of how he attempted to resolve each of these issues. Evaluation of the experiments carried out by Poutsma using this system (*op cit.*) indicate that, disappointingly, the Tree-DOT model performs poorly. The conclusions he draws suggest that larger-scale experiments on better quality data should yield a more accurate picture of the behaviour of the model. However, we find that his solutions to the main Tree-DOT implementation issues are not appropriate to these more demanding experiments. Thus, in the remainder of this chapter we focus on alternative solutions to these challenges so that a more detailed empirical evaluation of the Tree-DOT model can proceed.

5.1 Promising ideas, poor performance

The only implementation of a Tree-DOT system documented to date is that of Poutsma (2000, 2003). Here we provide details of his implementation and experiments and summarise his findings.

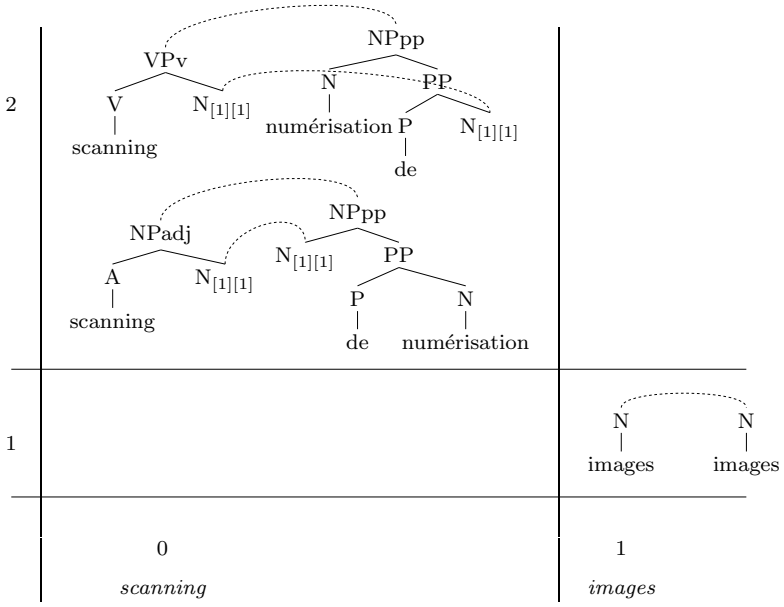


Figure 5.1: The Tree-DOT translation space for the input string *scanning images* given the Tree-DOT grammar in Figure 4.3(B).

5.1.1 Poutsma's Implementation

Computing the translation space The DOT *translation space* for any input string comprises all possible representations which can be assigned to that string according to the grammar. This space is very similar to the DOP parse space; the main difference is that each fragment comprises a pair of linked subtrees rather than a single subtree. Thus, as for DOP, we use a chart to store all fragments relevant to the input string, along with pointers to those fragments with which they can compose to form valid representations and, therefore, translations. An example DOT translation space is given in Figure 5.1. This translation space again comprises a two-dimensional chart of size N^2 where N is the length of the input string. Each token in the input string is assigned a number i such that $0 \leq i < N$. These numbers appear along the horizontal axis; the numbers which appear on the vertical axis (generally represented by j) indicate the number of input tokens spanned. Each open substitution site pair in every fragment present on the chart explicitly points to a chart position; any fragment composed at a substitution site must be selected from this position.

Poutsma adapts Bod (1998)'s approach to implementing Tree-DOP, described in section 2.4.1, in order to construct the DOT translation space for each input string. During

the analysis phase, each fragment is viewed as a rewrite rule where the left-hand side of each rule corresponds to the root node pair of a fragment and the right-hand side to the source and target frontiers of that fragment; this generic rule is shown in (5.1):

$$< root(t_s), root(t_t) > \longrightarrow < (frontier(t_{s_1}...t_{s_n})), (frontier(t_{t_1}...t_{t_n})) > \tag{5.1}$$

Each rewrite rule also has a pointer to the fragment which it represents, meaning that two identical rewrite rules which point to fragments with different internal structures remain distinct. Where frontiers are open substitution sites, the links between source and target sites are maintained. Thus, the topmost fragment in position [0][2] of the translation space in Figure 5.1 corresponds to the rewrite rule given in (5.2):

$$< VPv, NPpp > \longrightarrow < (\text{scanning}, \overset{\frown}{\text{N}}), (\text{numérisation}, \text{de}, \text{N}) > \tag{5.2}$$

This rewrite rule can be combined with rules that have the label $<N, N>$ on the left-hand side.

When fragments are viewed as rewrite rules in this manner, existing algorithms for context-free grammars can be applied to construct the derivation forest for a given input string. Poutsma (2000, 2003), however, gives no information as to which algorithm is used in his system, or whether or not it required adaptation to handle linked, bilingual rewrite rules.

The translation space contains all fragments which can be used to form a representation for the current input. Since every fragment in the derivation space comprises both a source and a target tree, each derivation read from this space automatically comprises both a source-language parse tree and a target-language parse tree. Consequently, generating a translation simply involves extracting the ordered frontiers from the target-language parse tree.

Selecting the best translation DOP parse probabilities are established by summing over the probabilities of the derivations yielding each parse. Similarly, DOT translation probabilities are established by summing over the probabilities of the derivations yielding

each source and target string. Thus, the problem of finding the most probable translation (MPT) for DOT is computationally analogous to the problem of finding the MPP for DOP and, again, exhaustive search of the translation space is not possible. Poutsma adopts the top-down breadth-first Monte Carlo sampling algorithm described by Hoogweg (2000). As the model requires maximisation of translation probability rather than parse probability, the frequencies of the translations in the sampled set should correspond to their DOT probabilities. However, while the analysis space built comprises paired source and target fragments, Poutsma (2003):347 states that "a random selection method to generate derivations from the target derivation forest" is used. As the DOT probability of sampling any fragment depends on both the source and target subtree root nodes of that fragment, sampling over target subtrees only means that the distribution of sampled translations is unlikely to correspond to their DOT distribution. Furthermore, Poutsma (*op cit.*) also states that "the random choices of derivations are based on the probabilities of the underlying subderivations" but does not discuss how these sampling probabilities are computed in his system. As discussed in section 2.5.1, there are several ways to compute sampling probabilities, each with different implications. Finally, Poutsma does not allow variation in the number of samples taken to reflect the level of translational ambiguity present in the input string: in every case, 1500 derivations are sampled and the most frequently-occurring translation in this set is returned.[1]

Pruning the fragment set Poutsma (2000, 2003) prunes the set of DOT fragments by varying maximum depth. As we will discuss in detail in section 5.2.1, however, it is not necessarily the case that the source and target subtrees in each fragment are of the same depth. As Poutsma does not address this issue, it is not clear whether he calculates fragment depth over the source subtree, the target subtree or by some other method.

[1]If each target linked node was transformed into a double category label comprising both the source and target node labels *and* sampling probabilities were calculated correctly, then correct trees and strings *could* be generated with the correct probabilities. However, from the description presented in (Poutsma, 2000, 2003), we cannot know exactly the properties of the sample set computed for each input string.

5.1.2 Experiments with the Verbmobil corpus

Poutsma carried out Tree-DOT experiments on a small section of the Verbmobil corpus consisting of 266 German-English sentence pairs. The dataset contained transliterated spoken appointment dialogues in German which were translated into English by native speakers of German. Each sentence was annotated with a context-free phrase structure tree and the tree pairs were aligned at both sentential and sub-sentential levels.[2] Poutsma (2000):55 states that the sentence alignments were not entirely correct as, sometimes, "two sentences in the German corpus corresponded to one sentence in English, or vice-versa." However, we do not agree that alignments should be considered erroneous simply because the correspondences between source and target sentences are not always 1-to-1. (Our approach to these alignments is given in section 6.1.) Furthermore, he says that he edited these incorrectly aligned tree pairs, but does not say *how* these pairs were edited. Finally, as the model cannot cope with unknown words, it was necessary to ensure that all test sentences could be analysed. Poutsma extended the training set by adding the smallest possible valid tree pairs into the training set; again, no information is given as to how many such fragments were inserted, nor as to how these fragments affected the frequency distributions. This dataset was split into three different training/test splits where each training set contained 226 tree pairs and each test set 40 sentences.

Translation experiments were then carried out using each of the three splits (and the results averaged) and translating both from German to English and English to German. These experiments were carried out using varying fragment depths from depth 1 through to depth 6 and finally using all possible fragments. However, as mentioned in section 5.1.1, no definition for fragment depth is given.

The results achieved are disappointing: exact match figures for experiments from English to German are 16%–19% and 13%–15% from German to English. Correct translation figures (i.e. counting translations which are either exact matches or well-formed alternatives) from English to German are 19%–24% and 18%–23% from German to English. In other words, at best one in every four translations was both grammatical and accurately preserved the meaning of the source string. Interestingly, we see little increase in transla-

[2]The sub-sentential alignments were inserted manually.

tion quality as the size of the fragments in the fragment base increases, meaning that the DOP Hypothesis is not confirmed for DOT.

5.1.3 Conclusions

As outlined in section 4.2, the Tree-DOT model embodies many desirable MT system characteristics and, consequently, would seem to be an MT paradigm worthy of investigation. Poutsma's pilot experiments yielded disappointing results, however, and he puts forward some possible explanations as to why this was the case (Poutsma, 2000):

- the dataset used was extremely small in size as it contained only 266 tree pairs; better performance is assumed if a larger corpus were to be used;
- the quality of the translations in the example base was poor as they were translated into English by non-native speakers and, as a result, the output translations were also poor;
- the trees in the dataset were wide and shallow rather than deep, meaning that varying tree depth did not have a great deal of impact on output quality, and Poutsma suggests that varying tree *width* would be more appropriate;
- it was frequently the case that two or more translations had roughly the same probability and the less preferred translation would have scored better;
- in the German language, case, number and gender influence the choice of word form and translation quality would improve if this information were integrated into the model.

We agree with Poutsma that the small number of sentence pairs and poor translation quality in his dataset contributed to the poor performance. However, it is also the case that the trees themselves were lacking in linguistic complexity. The total number of fragments yielded from 226 tree pairs in his dataset was 33479. In contrast, the 810 tree pairs contained in the English-French section of the HomeCentre corpus (which will be described more fully in section 6.1) yields a maximum number of fragments in excess of 250,000,000. In other words, although the number of tree pairs in Poutsma's dataset is

just under 28% of the number of pairs in the HomeCentre, it yields just 0.01% of the number of fragments. We suggest that, not only does the dataset need to be larger, but the analyses in the dataset need to provide a greater level of linguistic detail in order to fully assess the capabilities of the model.[3] We agree with Poutsma that incorporating information on features such as number and gender into the model is likely to significantly improve translation quality. This issue is discussed further in chapter 8. However, we also feel that the Tree-DOT model merits further investigation and evaluation before moving on to more linguistically-complex models.

Implementing the Tree-DOT model is analogous to implementing Tree-DOP. Given the discussion in sections 2.4 and 2.5.1 on the difficulties of efficiently implementing the Tree-DOP model, however, it is clear that Poutsma's implementation lacks the sophistication to be able to handle a dataset which constitutes a significant increase in terms of size and complexity on the dataset used previously. Furthermore, it is possible that in the situations Poutsma mentions where two or more translations had roughly the same probability and the less preferred translation would have scored better, this is at least partly attributable to the inadequacies of his sampling methodology.

In conclusion, the algorithms developed to make Tree-DOP more efficient must be adapted for Tree-DOT and a more robust implementation built to facilitate experiments using larger, more complex datasets. Until such a system is in place and these experiments carried out, it will not be possible to fully demonstrate the merits of the Tree-DOT approach to translation.

[3]Much of Poutsma's discussion as to the quality of translations produced by the DOT model centers around manual comparison with the translations output by the Systran MT system for the same sets of test sentences. Clearly, the poor quality of the translations in Poutsma's training data meant that the DOT model yielded poor translations compared to those generated by Systran. Consequently, we agree that poor training translation quality contributed to the disappointing evaluation outcome. However, we do not feel that this is the most appropriate way to evaluate data-driven MT systems, and that automatic evaluation gives a better picture of how such models perform. If a data-driven system is trained on poor quality translations, then we expect to get poor translations out, but if the reference translations are also poor then we still expect to score well. In a data-driven system, the aim is to model the data supplied, and if evaluation is over a held-out portion of this data then we expect to *score* well even if the output translations are, in human terms, of poor quality. Although he also performed automatic evaluation against reference translations using a metric he defined himself (Poutsma, 2000):58, Poutsma's work predates the development of the automatic evaluation metrics – Bleu (Papineni et al., 2001, 2002), NIST (NIST, 2002; Doddington, 2002) and F-score (Melamed et al., 2003; Turian et al., 2003)) – currently in use. Poutsma's metric, termed 'Largest Translation Part', is far less sophisticated than these newer metrics, and so it is difficult to draw meaningful conclusions from the automatic evaluation he presents.

5.2 A new implementation of the Tree-DOT model

As addressed in chapter 2, the main issues when building an implementation of the Tree-DOP model are how to prune the fragment set, how to build the DOP parse space for the input string and how to establish which parse in the parse space is best for that string. Exactly the same issues arise when implementing the corresponding Tree-DOT translation model. The major practical differences between Tree-DOP and Tree-DOT implementations come down to the facts that (i) each fragment comprises a linked pair of subtrees rather than a single subtree and (ii) we wish to search for the most probable string rather than the most probable parse tree. Thus, in this section we discuss the options for DOT grammar induction, translation space computation and output translation ranking.

5.2.1 Pruning the fragment space: link depth

The refinement of the fragmentation process to account for translational links may (and often does) result in a smaller number of DOT fragment per tree pair than would be the case with DOP. Recall that, as given in section 2.3, the number of monolingual DOP fragments $F_{DOP}(A_T)$ projected from non-terminal node A_T in treebank tree T which has children $C_T = \{C_{T_1}...C_{T_n}\}$ is calculated according to equation (5.3).

$$F_{DOP}(A_T) = \prod_{C_{T_x} \in C_T} (F_{DOP}(C_{T_x}) + 1) \tag{5.3}$$

Recall also that the total number of fragments $TF_{DOP}(T)$ which can be extracted from treebank tree T is the sum over the number of fragments which can be projected from each of its nodes, as stated in equation (5.4).

$$TF_{DOP}(T) = \sum_{A_T \in T} F_{DOP}(A_T) \tag{5.4}$$

Consider, for example, the English tree on the left-hand side of the Tree-DOT representation given in Figure 5.2.[4] According to equations (5.3) and (5.4), this tree yields 357

[4]While English and French are considered to be syntactically similar languages, in certain contexts they exhibit strong stylistic divergences. In this particular translation example, the English printer manual section header is phrased as a question, whereas the corresponding French translation of that header is realised as a declarative sentence. We provide further discussion regarding translational divergences

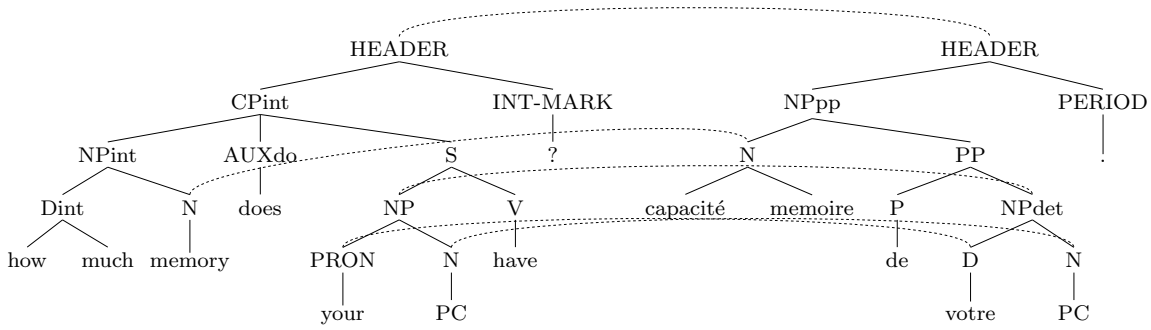


Figure 5.2: Tree-DOT representation.

monolingual DOP fragments. Furthermore, the French tree on the right-hand side of the Tree-DOT representation given in Figure 5.2 yields 87 fragments according to these equations.

The definitions of the *root* and *frontier* operations given for Tree-DOT in section 4.2.1, which are used to extract DOT fragments from linked tree pairs, distinguish between linked and unlinked nodes. Thus, calculating the number of bilingual fragments extracted from each tree pair also requires that we distinguish between linked and unlinked nodes. Accordingly, the number of bilingual DOT fragments $F_{DOT}(A_T)$ projected from non-terminal node A_T in (source or target) treebank tree T which has children $C_T = \{C_{T_1}...C_{T_n}\}$ is calculated according to equation (5.5).

$$F_{DOT}(A_T) = \prod_{linked(C_{T_x}) \in C_T} (F_{DOT}(C_{T_x}) + 1) \prod_{unlinked(C_{T_x}) \in C_T} F_{DOT}(C_{T_x}) \quad (5.5)$$

Note that, for each linked tree pair, applying equation (5.5) to any linked node in the source tree and to *the node to which this source node is linked* in the target tree yields exactly the same result. For example, applying this formula at the root node of the English tree in Figure 5.2 (labelled HEADER) and the root node of the French tree (labelled HEADER) to which it is linked indicates that 10 subtrees are projected from each. Furthermore, each of the 10 English subtrees corresponds to one of the 10 French subtrees, meaning that a total of 10 bilingual paired subtrees (i.e. DOT fragments) are projected from the node pair <HEADER,HEADER> of this tree pair. The total number of fragments $TF_{DOT}(T)$ which can be extracted from each pair of linked trees T is the sum over the number of

occurring in the data we use to train and test the DOT model in section 6.1.1.

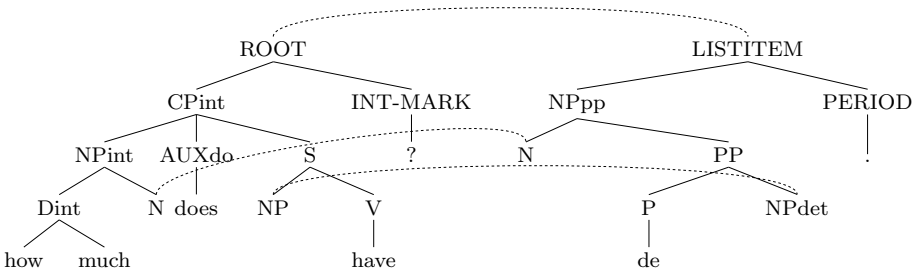


Figure 5.3: Tree-DOT fragment extracted from the representation in Figure 5.2.

fragments which can be projected from each of either the source or target tree's linked nodes, as stated in equation (5.6).

$$TF_{DOT}(T) = \sum_{linked(A_T) \in T_{s|t}} F_{DOT}(A_T) \tag{5.6}$$

Thus, according to equations (5.5) and (5.6), the number of DOT fragments which can be extracted from the linked tree pair in Figure 5.2 is 17.

As we model translational dependencies rather than monolingual dependencies, the number of DOT fragments extracted from a linked tree pair is generally less than the number of DOP fragments which can be extracted from each of the source and target trees comprising that tree pair. We have already illustrated this above: the source tree in Figure 5.2 yields 357 monolingual fragments and the target tree 87 monolingual fragments but the bilingual tree pair yields just 17 DOT fragments. Nevertheless, pruning methods to constrain the size of the fragment base are still necessary. We discussed the relative merits of several pruning methods for Tree-DOP fragments in section 2.3. These methods involve excluding fragments on the basis of fragment properties such as depth, number of lexicalised frontiers, number of non-headword frontiers and number of open substitution sites. The only one of these pruning criteria which can be applied directly to the DOT fragment base is the restriction on the number of open substitution sites per fragment.

As, in every DOT fragment, each source non-terminal frontier node is linked to exactly one target non-terminal frontier node and vice versa, the source and target subtrees in each fragment always have the same number of open substitution sites. Thus, the number of open substitution sites in a fragment can be calculated as the number of links between non-terminal frontier nodes in that fragment, and placing a restriction on this

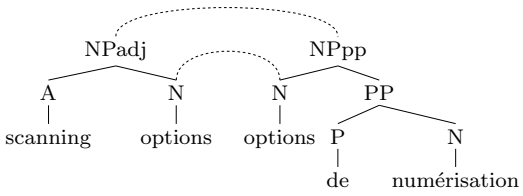


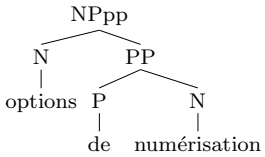
Figure 5.4: Tree-DOT fragment.

fragment property is straightforward. As pointed out in (Way, 2001):187, however, it is not necessarily the case that the source and target subtrees in each fragment have the same number of terminal frontier nodes. For example, although the fragment in Figure 5.3 has exactly 2 open substitution sites in each subtree, the source subtree has 4 terminal frontiers[5] whereas the target subtree has only 2. Thus, calculation of the number of terminal frontiers in a fragment involves making a decision as to whether source or target terminal frontiers should be counted. Way (*op cit.*) also observes that if, for example, the number of terminals is counted on the source subtree and the maximum is set to 3 then fragments such as the one given in Figure 5.3 will be excluded. He suggests that manual intervention may be necessary to prevent such fragments from being pruned. Furthermore, we note that pruning the fragment base by placing an upper limit on fragment depth is also problematic for DOT as fragments such as the one given in Figure 5.4 do not necessarily comprise source and target subtrees of the same depths. Again, use of depth restrictions directly as they are used for DOP involves making an arbitrary decision as to whether source or target depth should be calculated. However, we observe that this issue is merely a surface symptom of the fundamental difference between the dependencies modeled by the DOP and DOT fragment sets rather than constituting a problem in itself: DOP models monolingual dependencies whereas DOT models bilingual dependencies.

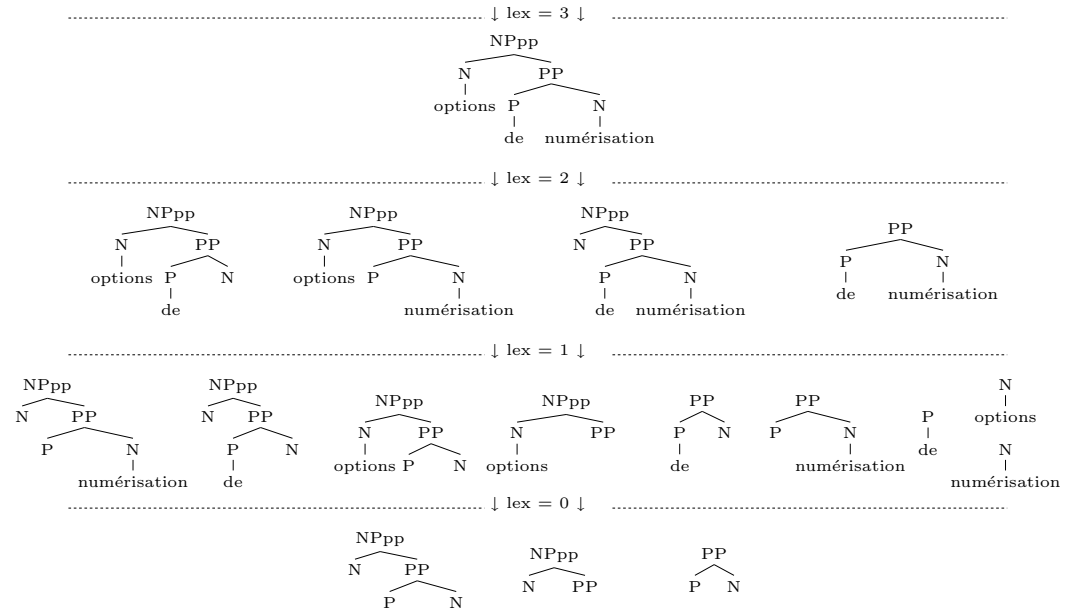
As discussed in section 2.3, the full set of DOP fragments captures all arbitrary dependencies occurring in a given training treebank. Use of pruning techniques reduces this set such that only a subset of the dependencies present are actually captured. Although this subset may be specified over quantitative rather than linguistic characteristics of the full fragment set, it is nevertheless the case that the choice of dependencies modeled is no longer arbitrary.

[5] As the first two words of the source string – *how much* – share the same parent node, they are treated as a single lexical unit.

(A) A phrase-structure tree representing the French string *options de numérisation*:



(B) Organisation of the DOP fragments extracted from (A) according to the number of frontier terminals in each:



(C) Organisation of the DOP fragments extracted from (A) according to the depth of each:

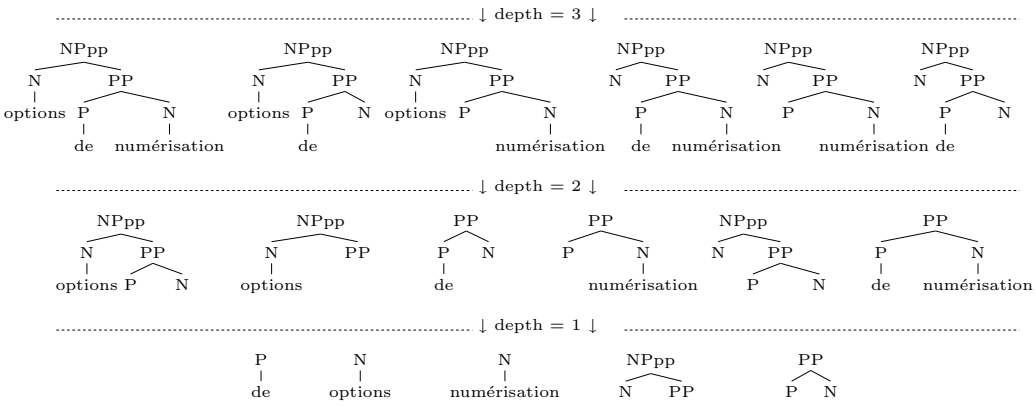
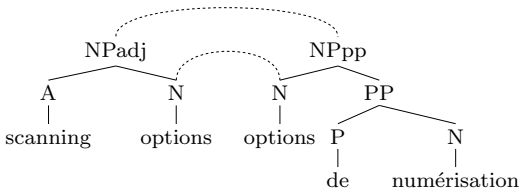


Figure 5.5: 17 unique DOP fragments can be extracted from the phrase-structure tree in (A). (B) shows these 17 fragments organised according to the number of terminal frontier nodes in each and (C) shows these same 17 fragments organised according to fragment depth.

Consider, for example, the treebank tree given in Figure 5.5(A). This tree yields the 17 unique DOP fragments given in Figure 5.5(B) and (repeated in) (C). In Figure 5.5(B), these fragments are organised in terms of how many lexicalised frontiers each fragment has, and in (C) they are organised according to fragment depth. Looking firstly at Figure 5.5(B), we see that if we prune the fragment base by excluding all lexicalised fragments, then just 3 fragments remain. These fragments model structural dependencies only. Clearly, in this case, the input string must be tagged before it can be parsed with this DOP grammar. If we relax the restriction to allow fragments with maximally one lexicalised frontier then 12 fragments are included in the fragment base. The fragment base now provides varying amounts of information about the structural contexts in which each terminal can occur. Relaxing the restriction further to incorporate fragments with maximally two lexicalised frontiers allows us to model bilexical dependencies also. However, the set of fragments comprising maximally one lexicalised frontier encodes information about *all* the terminals in the treebank tree in Figure 5.5(A). Looking at Figure 5.5(C), we see that including only fragments of depth 1 in the fragment base restricts us to capturing local dependencies. However, as is the case when we allow at most 1 terminal per fragment, information about every terminal in the treebank tree is encoded at depth 1. Thus, adding fragments of increasingly greater depths allows us simply to capture more and more probabilistic information about the lexical items already present in the fragment base.

Clearly, the effects of applying pruning thresholds to the DOP fragment base are predictable. In particular, we know that – with the exception of restricting to unlexicalised fragments only, which is not generally done in practice – the minimum amount of information encoded about each word in the treebank is its part-of-speech tag. If we apply these same pruning techniques to the DOT fragment space by calculating fragment properties over either the source or target subtrees, however, the effects on the dependencies modeled are *not* predictable. In particular, if we proceed using this methodology then we can no longer be sure, as we were for DOP, that there is some minimal amount of information encoded about each word in the treebank. This is due to the fact that the DOT fragment base captures *translational* lexical and structural dependencies rather than monolingual

(A) A linked tree pair:



(B) The set of DOT fragments extracted from the linked tree pair in (A):

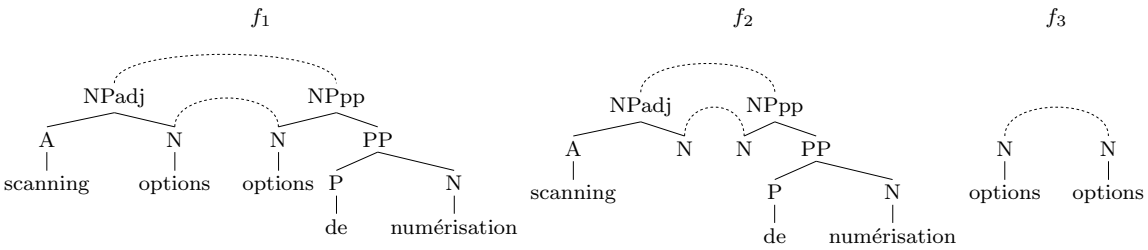


Figure 5.6: The linked pair of trees given in (A) yields the set of DOT fragments given in (B).

ones. Consider, for example, Figure 5.6, where the linked pair of trees given in (A) yields the set of DOT fragments given in (B). (Note that the target subtree in (A) is exactly the DOP representation provided in Figure 5.5.) If we restrict the fragment base such that it includes fragments of depth 1 only, then regardless of whether we measure depth over the source or target subtrees, the fragment base will comprise fragment f_3 only. Thus, by omitting all other fragments we retain no information about the English word *scanning* and the French words *de* and *numérisation*. Similarly, if we restrict the fragment base such that it includes fragments with maximally 1 target terminal frontier then fragment f_3 will again be the only fragment remaining. (While calculating degree of lexicalisation over the source subtrees will, in this particular instance, result in the retention of fragment f_2 also, this is by no means predictable.)

In section 4.2.1 we saw that the DOT fragmentation operations work over linked nodes only. Correspondingly, in order to calculate the number of fragments yielded by each DOT representation, we differentiate between linked and unlinked nodes (equations (5.5) and (5.6)) as linked nodes are productive whereas unlinked nodes are not. Accordingly, here we conclude that direct application of DOP pruning methods to the DOT fragment base is inappropriate because no distinction is drawn between linked and unlinked nodes.

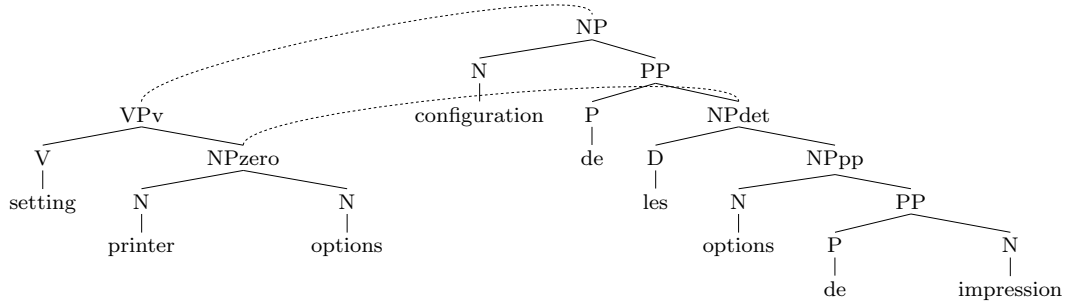


Figure 5.7: sourcedepth = 3, target depth = 6, *link depth* = 2

Consequently, we replace the notion of fragment depth – the greatest number of steps taken to get from the root node to any frontier node – with the notion of *link depth* for fragments comprising linked subtree pairs (Hearne and Way, 2003). The link depth of a fragment is the greatest number of steps taken *which depart from a linked node* to get from the root node to any frontier node. This yields the same result whether calculated over the source or target side of the fragment. For example, for the fragments comprising two subtree pairs given in Figure 5.7, the depth of the source language subtree (on the left) is 3 whereas the depth of the target language subtree is 6. If, however, we simply calculate the depth of the fragment as a whole using the concept of link depth, we arrive at fragment depth of 2.

Consider again the fragment set in Figure 5.6(B). According to the definition of *link depth*, both fragments f_2 and f_3 are of depth 1, meaning that the minimal fragment set comprises these two fragments only. Clearly, these two fragments encode the minimum amount of information about each word in the treebank as each word is contained in one of these fragments. Fragment f_1 is of link depth 2 and adds (only) further structural and contextual information about the words already contained in the fragment base. Thus, not only does link depth characterise each bilingual fragment as a whole, but pruning the DOT fragment base according to link depth changes the dependencies occurring in the fragment base in a predictable way. Henceforth, this is the method we use to calculate DOT fragment depth.

5.2.2 Translation-space construction

As discussed in section 5.1.3, we do not feel that the translation algorithm proposed by Poutsma (2000, 2003) – which is based on the 'fragments as rewrite rules' technique proposed for DOP by Bod (1992) and described in section 2.4.1 – will facilitate the experiments required to fully assess the performance of the DOT model. Thus, in this section we focus on the adaptation of more efficient DOP parsing algorithms to the DOT translation model. Firstly, we discuss how the fragments in the DOT translation space for an input string relate to the fragments in the DOP parse space for that same string. In light of this relationship, we then outline the elements which building the parse and translation spaces have in common, and give the general intuition as to how the former can be used in creating the latter. Finally, we discuss in detail the possibility of adapting the DOP parsing algorithms developed by Goodman (1996a, 1998, 2003) and Sima'an (1995a, 1999) to accomplish the task of translation space computation.

From parsing to translation: the general model

Conceptually, the source- and target-language halves of each DOT fragment, along with the translational links between them, form a single unit. It is useful on a practical level, however, to make explicit the relationships between (i) the two halves of the set of bilingual DOT fragments which can be extracted from a set of linked training trees and (ii) the two sets of monolingual fragments which can be extracted from that same set of linked training trees by placing the source and target trees in separate sets, discarding the links and applying the DOP fragmentation operations. In other words, if one of the languages represented in the bilingual treebank is language L, what is the relationship between (i) the fragment set F_b generated by applying the DOT fragmentation operations to the bilingual treebank and then stripping away the links and corresponding-language parts of each extracted fragment, leaving only representations for L and (ii) the fragment set F_m generated by taking the bilingual treebank, stripping away the links and corresponding-language trees and applying the DOP fragmentation operations to this monolingual treebank? As the fragmentation operations defined for Tree-DOT can only select linked nodes to be either root or frontier nodes, it follows that non-linked nodes are always internal to the fragments

in which they occur. Thus, set F_b comprises a subset of the fragments in F_m such that all root nodes and substitution sites of fragments in F_b are linked to target-language nodes in the bilingual treebank.

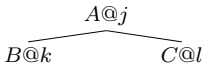
In Tree-DOT, the process of building the translation space is driven by the input string, and the building of target language representations can be viewed as a by-product of parsing with bilingual fragments. It is possible, therefore, to build a first approximation of the translation space by simply parsing with the source-language half of the bilingual fragment base only, i.e. fragment set F_b. Once this has been accomplished, the one or more target-language subtrees which correspond to each source-language fragment in the approximated space are retrieved. However, according to the DOT composition operation, the target-language subtrees effectively act as constraints on the source-language fragments which can combine to form analyses: in order for fragment f_x with root node categories $<R_{s_x},R_{t_x}>$ to compose with fragment f_y with leftmost substitution site categories $<LSS_{s_y},LSS_{t_y}>$, not only must the source root in f_x, R_{s_x}, correspond to the source leftmost substitution site in f_y, LSS_{s_y}, but the target root R_{t_x} must also correspond to the substitution site category LSS_{t_y}. Effectively, this means that source-language fragments which can combine freely in a monolingual model are now constrained by their target-language links. Thus, fragments which, due to translational constraints, cannot be composed with any other fragments to form valid analyses are removed from the approximated space, giving us the bilingual parse and translation space for the input string.

When the task of building the DOT translation space is viewed from this perspective, adaptation of the parsing algorithms of Goodman (1996a, 1998, 2003) and Sima'an (1995a, 1999) to accomplish this task seems worthy of investigation. However, we find that Sima'an's two-phase analysis method gives the required flexibility whereas Goodman's PCFG-reduction method does not. In the remainder of this section, we detail why this is the case.

Translating with Goodman's PCFG-reduction approach

Recall that, as described in section 2.4.3, Goodman (1996a, 1998, 2003)'s algorithm for computing the DOP parse-space for an input string reduces the DOP fragment set to

a PCFG containing maximally 8 rules for each node in the training treebank. Each training-tree node A is assigned a unique address k and, correspondingly, one new non-terminal node A_k is created; such non-terminals are called "interior" nodes and the original nodes "exterior" nodes. In addition, the number of subtrees a_k with root node A_k is also calculated.



(5.7)

For any node grouping such as the one in example (5.7), the eight PCFG rules and their corresponding probabilities in example (5.8) are then extracted.

$$
\begin{array}{llllllllll}
(1) & A_j & \longrightarrow & BC & (\frac{1}{a_j}) & (5) & A & \longrightarrow & BC & (\frac{1}{a}) \\
(2) & A_j & \longrightarrow & B_kC & (\frac{b_k}{a_j}) & (6) & A & \longrightarrow & B_kC & (\frac{b_k}{a}) \\
(3) & A_j & \longrightarrow & BC_l & (\frac{c_l}{a_j}) & (7) & A & \longrightarrow & BC_l & (\frac{c_l}{a}) \\
(4) & A_j & \longrightarrow & B_kC_l & (\frac{b_kc_l}{a_j}) & (8) & A & \longrightarrow & B_kC_l & (\frac{b_kc_l}{a})
\end{array}
\tag{5.8}
$$

These rules correspond to the eight possible contexts in which the node grouping in example (5.7) can occur in fragments extracted from the corresponding treebank tree; each of the three nodes can be either interior or exterior (i.e. root node or substitution site) to any fragment in which the grouping occurs. Thus, every relevant DOP fragment can be constructed using one or more PCFG derivations by converting each internal node to an external node and, furthermore, the probability of each of these DOP fragments can be calculated by summing over the PCFG derivations yielding that fragment.

This is a very attractive algorithm for DOP as the size of the extracted PCFG is far smaller than the corresponding fragment set and because looking back to the fragment set is not necessary. However, the inflexibility of this approach – discussed in detail in section 3.1.1 – makes it unsuitable for use in a DOT system on several levels. Importantly, the advantage of not having to look back to the fragment base has, in the context of translation, turned into a disadvantage: it is extremely computationally expensive to look back to the fragment base in situations where that becomes necessary.

As the set of source-language DOT fragments is simply a subset of the corresponding DOP fragment set such that certain treebank tree nodes are not permitted to be external, Goodman's PCFG reduction method can also be used to characterise the source-language parse space for the input string over the set of bilingual fragments. In order to achieve

this, we simply extract the PCFG rules from the source side of the bilingual fragment set subject to the restriction that rules specifying that an unlinked node is external are not generated. If, for example, in the node grouping given in example (5.7) only nodes $A@j$ and $B@k$ were linked and, consequently, node $C@l$ was never external to a fragment then only rules 3, 4, 7 and 8 from example (5.8) would be extracted.

As well as using the PCFG reduction to characterise the subtree structures relevant to the input string, it must also characterise the parse space *probabilistically*. In other words, the rule probabilities must also be estimated such that the probability of deriving each valid fragment is equal to its relative frequency in the DOT fragment base. As it stands, the rule probabilities given correspond to the frequency distribution of the source-language subtrees in the bilingual fragment base rather than the frequency distribution of the source and target subtree pairs. We can augment each linked source-language subtree node with the category of the target-subtree node to which it is linked. For example, source-language node NP linked to target node PP would be assigned the category label NP.PP; this 'category' would thus be distinct from, for example, source-language node NP linked to target node NP which would be labelled NP.NP. As DOT fragment probabilities are conditioned on root node pairs, this transformation allows us to correctly establish the counts for the number of subtrees headed by each root node pair. (The counts for subtrees whose root nodes are internal to the source-language fragment are calculated as for DOP.)

However, we see no way of adapting this PCFG reduction so that the target-language subtrees are also characterised. At best, we could use the PCFG space to rebuild each source-language subtree and recover its target-language counterpart by matching it against the training data. However, this involves explicitly recreating every fragment relevant to the input string which, in turn, requires that we prune the fragment set. As discussed in section 3.1.1, pruning the fragment set so that the parse space is computable unfortunately results in a large increase in the size of the PCFG-reduction (if, indeed, it is even possible to compute the corresponding PCFG-reduction) and this algorithm loses its advantage. Thus, we do not use Goodman (1996a, 1998, 2003)'s PCFG-reduction method in our DOT implementation.

Translating with Sima'an's two-phase analysis approach

As described in section 2.4.2, Sima'an (1995a, 1999)'s two-phase analysis approach takes the context-free grammar underlying the fragment set and uses it to approximate the parse space of the input string. Correspondences between these CFG rules and the fragments in which they occur then facilitate the transition from this CFG parse space to the required DOP parse space for the input. The underlying CFG is, however, non-probabilistic; fragment probabilities are estimated by looking back to the full fragment set. This algorithm can be applied to the computation of the DOT translation space for a given input string in a very straightforward manner.

Each DOT fragment is associated with a unique identifier. The CFG underlying the source side of the fragment set is extracted such that each rule in the CFG is associated with the set of fragment identifiers in which it occurs. The two-phase analysis algorithm is then applied exactly as for DOP, as described in section 2.4.2. This algorithm generates a monolingual parse space comprising those source-subtrees which can be used to parse the input string. However, as we also retain the fragment identifiers of each of these source-subtrees, recovering the translational counterpart of each subtree, as well as the DOT probability of the fragment as a whole, is trivial. Finally, fragments which, due to translational constraints, cannot be composed with any other fragments to form valid analyses are removed from the approximated space, giving us the bilingual parse and translation space for the input string. As we discuss in section 5.2.4, several different disambiguation strategies can now be applied to this translation space in order to select the best translation to output.

5.2.3 Compact fragment representation

Explicitly creating the DOP fragment base is expensive due to the very large numbers of fragments that must be extracted, counted, stored and compiled. As the two-phase algorithm used to compute the parse space for each input string requires only an indication as to which fragments each underlying CFG rule appears in, it is not necessary to explicitly extract and store the fragment set. Thus, in section 3.1.2 we introduced a dynamic method to establish the fragment set on the fly such that only the treebank trees themselves need to

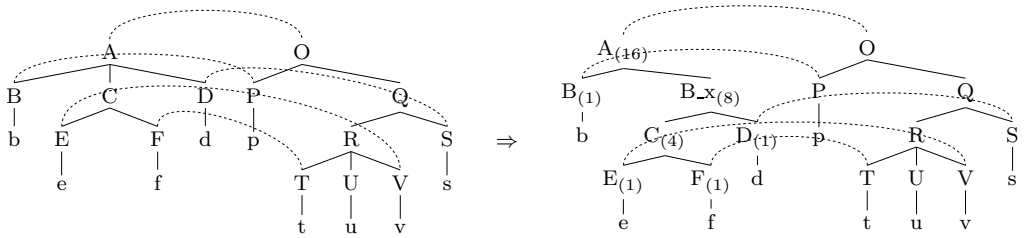
be stored. The same issues with regard to fragment set extraction arise for DOT. However, the expense of storing and compiling the DOT fragment set is even greater because each fragment now comprises two subtrees, along with the links between them. Fortunately, our on-the-fly fragment set extraction can also be applied to bilingual linked treebanks in a straightforward manner. Explicit fragment characterisation is done over source trees only and the target subtrees retrieved when converting from the monolingual derivation space to the bilingual derivation space.

We first apply the DOT root operation to each of the paired treebank representations, yielding a set of 'intermediate' fragments as for DOP but, this time, the size of this set is linear in the number of *linked* node pairs in the treebank. The DOT frontier operation is then applied by assigning to each node n in the *source* side of each intermediate fragment a set of fragment identifiers such that if its left and right child nodes n_l and n_r are present in a fragment then the corresponding fragment identifier appears in the node's identifier set. Either both n_l and n_r are present in the fragment or neither are present, in which case node n is itself either a substitution site or not in the fragment. Thus, the presence of fragment identifier f_{id} at node n in the source subtree signifies that the CFG rule $n \longrightarrow n_l\ n_r$ occurs in the source side of fragment f_{id}.

When assigning DOT fragment identifiers to each node in each source subtree, we must again account for the distinction between linked and unlinked nodes. Recall that, for DOP, ECNF nodes of the form X_y (which are inserted into the treebank trees during conversion to binary format) never occur as either root or frontier nodes because they must always be internal to those fragments in which they appear. In fact, unlinked nodes in DOT fragments can be treated in the same way as these ECNF nodes as they also must always be internal to those fragments in which they occur.

We partition the set of identifiers at source node n with left and right child nodes n_l and n_r into four sets representing the four possible combinations of internal and external child nodes $<n_{l_s},n_{r_s}>$, $<n_{l_s},n_{r_i}>$, $<n_{l_i},n_{r_s}>$ and $<n_{l_i},n_{r_i}>$. However, if node n_l is unlinked then sets $<n_{l_s},n_{r_s}>$ and $<n_{l_s},n_{r_i}>$ remain empty as this node is never a substitution site. Similarly, if node n_r is unlinked then sets $<n_{l_s},n_{r_s}>$ and $<n_{l_i},n_{r_s}>$ are empty, and if both child nodes are unlinked then the only non-empty set is $<n_{l_i},n_{r_i}>$. This is illustrated

(A) Root-generated 'intermediate' fragment whose source subtree has been converted to ECNF (through which source node B_x has been inserted) and each source node annotated with the number of different fragments yielded through application of the frontier operation:



(B) Source node annotations representing all possible frontier operations where the total number of frontier operations possible is 16 and the fragments corresponding to each of these frontier operations have been allocated identifiers from the set of integers 1 - 16:

$A_{(16)}$	<B_s,B_x_s>:{} <B_s,B_x_i>:{1-8} <B_i,B_x_s>:{} <B_i,B_x_i>:{9-16}
$B\_x_{(8)}$	<C_s,D_s>:{} <C_s,D_i>:{} <C_i,D_s>:{1-4,9-12} <C_i,D_i>:{5-8,13-16}
$C_{(4)}$	<E_s,F_s>:{1,5,9,13} <E_s,F_i>:{2,6,10,14} <E_i,F_s>:{3,7,11,15} <E_i,F_i>:{4,8,12,16}
$B_{(1)}$	<b>:{9-16}
$E_{(1)}$	<e>:{3,4,7,8,11,12,15,16}
$F_{(1)}$	<f>:{2,4,6,8,10,12,14,16}
$D_{(1)}$	<d>:{5-8,13-16}

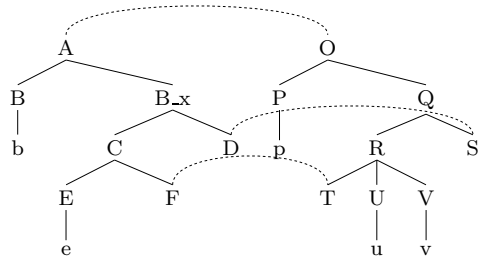
Figure 5.8: The 'intermediate' fragment in (A) was generated by the root operation. (B) gives the source node annotations representing all possible frontier operations where the total number of frontier operations possible is 16 and the fragments corresponding to each of these frontier operations have been allocated identifiers from the set of integers 1 - 16.

by the example in Figure 5.8 where the node annotations in (B) correspond to the DOT 'intermediate' tree in (A). Consider, for example, the annotation for node B_x. Its left child node, C, is an unlinked node whereas its right child node, D, is a linked node. Thus, the annotation sets specifying node C as a substitution site are empty.

Extracting these partitioned sets of fragment identifiers along with each source-language CFG rule extracted gives us the correspondence between the source side of the fragment set and this CFG. Thus, we can transition from phase 1, in which the source-CFG space is constructed, to phase 2, thereby generating the monolingual space comprising those source-subtrees which can be used to analyse the input string. For DOP, we stated that retrieval of any fragment can be accomplished easily by simply checking for its absence or

presence, as an internal node or substitution site, at each node in the intermediate tree. Although this is not strictly necessary for DOP parsing as these fragments are reconstructed automatically using the annotated CFG during phase 2, it is crucial for DOT as it allows us to retrieve the target-language subtree corresponding to each source-language subtree in the derivation space.

Essentially, the set of nodes identified as open substitution sites in any source subtree also characterise its linked target-language counterpart. Consider, for example, the situation where the source subtree of fragment f_{11} in Figure 5.8 is relevant to the input string, and so we wish to retrieve the corresponding target subtree. We first look at the retrieval of the source subtree. The sets corresponding to node A indicate that nodes B and B_x are both internal to fragment f_{11}. Trivially, this also means that terminal symbol b is a frontier node. The sets corresponding to node B_x signify that while node C is internal to f_{11}, node D is a substitution site. Finally, the annotation at C indicates that node E is internal to f_{11} (and so e is a frontier terminal symbol) whereas node D is a substitution site. Thus, we see that fragment f_{11} comprises two open substitution sites labelled D and F. Recall that, in every DOT fragment, each source non-terminal frontier node is linked to exactly one target non-terminal frontier node and vice versa. Consequently, the only possible open substitution sites in the target subtree are those linked to the source nodes labelled D and F. Looking at the target side of the intermediate tree, we conclude that the target subtree of fragment f_{11} comprises all nodes in the intermediate tree except those dominated by the nodes linked to the source open substitution sites D and F, which are, themselves, target open substitution sites. Thus, fragment f_{11} corresponds to the fragment given in example (5.9). For disambiguation purposes we retain the links between source and target root nodes and open substitution sites, but all internal links can be discarded.



(5.9)

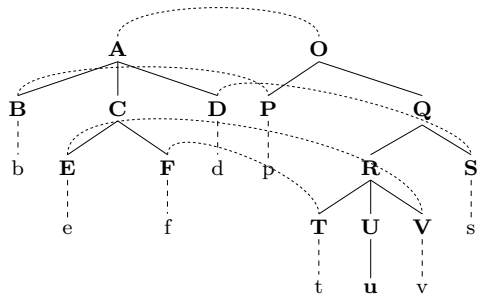
As we complete each DOT fragment in the derivation space, we ascertain that there

exists at least one fragment in the derivation space with which each of its (paired) open substitution sites can potentially be composed. If, for any such site, no fragments are found then this fragment is discarded as it cannot be used in building a DOT derivation for the input string. Thus, completion of this process yields the required bilingual parse and translation space for the input string.

Calculating relative frequencies from compact fragment representations

As for DOP, calculation of relative frequencies (and removal of identifiers corresponding to duplicate fragments) over these compact bilingual fragment representations is straightforward.

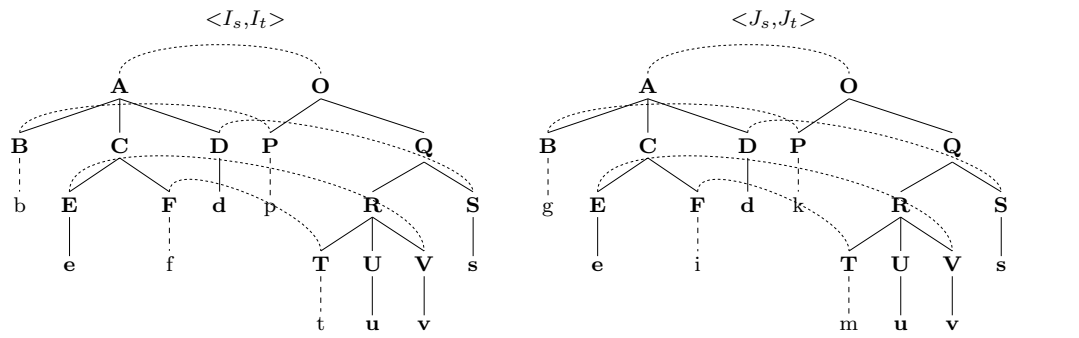
Two intermediate fragments $<I_s,I_t>$ and $<J_s,J_t>$ encode duplicate DOT fragments if connected portions of those fragments which start at their root node pair are identical. Minimally, these connected portions must comprise the fragment of link depth 1 which can be projected from the intermediate fragment. An example of such a minimal portion is given in (5.10), where the solid lines connect the nodes forming part of the minimal portion and the dashed lines denote nodes outside this portion.



(5.10)

The extension of the portions of intermediate fragments which yield identical DOT fragments is a recursive process: for each linked node already in the identical portion of each fragment, check that every path from this node to either another linked node or a terminal frontier node has a corresponding identical path in the fragment it is being compared to. In example (5.11), the identical portions of intermediate fragments $<I_s,I_t>$ and $<J_s,J_t>$ can be extended to include the paths from linked node pair $<E,V>$ to frontiers e and v and from linked node pair $<D,S>$ to frontiers d and s because these terminal frontier nodes occur in both fragments. However, as the children of the node pairs

$<B,P>$ are different in each of the fragments we are comparing, the identical portions cannot be extended to include these children; this is also the case for the node pairs labelled $<F,T>$.



(5.11)

Once we have identified the nodes included in the identical fragment portions, we have established exactly the set of fragments which are identical to each other: as for DOP, all boundary-identical nodes (i.e. those nodes which are included in the identical portions but whose children are not) are either substitution sites in the duplicate fragments or do not occur in those fragments. When the identifiers of these fragments have been established, we increment the counts of those fragments in one of the trees being compared and delete the identifiers of those fragments from the node annotations of the other.

5.2.4 Ranking translations

As discussed in section 5.2.2, adapting Sima'an (1995a, 1999)'s two-phase analysis approach for DOT allows for flexibility with regard to choice of disambiguation strategy; this was also the case for parsing, as discussed in section 3.1.1. In this section we discuss the application of the sampling techniques of Chappelier and Rajman (2003) to the search for the most probable translation for the DOT model. We also discuss the possibility of selecting for output the translations yielded by the most probable representation, the most probable derivation and the shortest derivation.

Most Probable Translation

The DOT model calls for ranking of the output translations according to their probabilities; this requires summing over the probabilities of all derivations which yield each target string

and, therefore, is computationally analogous to the calculation of parse probabilities for DOP. We approximate the search for the most probable translation according to the DOT model by applying the random sampling algorithm used for DOP. Again, we sample a derivation by selecting and composing fragments at random until no open substitution sites remain. Here, however, each fragment comprises a linked source and target subtree pair, meaning that each sampled derivation also comprises a linked tree pair such that the ordered sequence of terminals of the target tree constitutes a valid translation of the input string according to the evidence presented in the training data.

Recall that, as discussed in section 2.5.1, for DOP our objective for calculating the *exact* sampling probability of each fragment is to ensure that the frequency of each parse tree in the sampled set corresponds to its DOP probability conditioned on the input string when the sample set is large enough. Thus, we calculate bottom-up the sampling probability of each fragment in the parse space as its DOP probability by the total sampling probability mass available at each of its substitution sites. We employ precisely the same strategy when sampling DOT derivations, taking into account the fact that each substitution site comprises a linked category pair rather than a single category. Just as DOP parse probability is calculated by summing over derivation probabilities, DOT translation probability is also calculated by summing over derivation probabilities. Accordingly, calculating the exact sampling probability of each DOT fragment and using these probabilities to determine the likelihood of selecting random derivations means that, when a sufficiently large number of samples have been taken, the frequency distribution of the translations yielded by the sampled derivations corresponds to their DOT probability distribution. Thus, the most frequent translation in the sampled set is also the most probable translation according to the Tree-DOT model.

We also adopt the BKS method – as described in section 2.5.1 – to determine, for each input string, when enough samples have been taken. We restate the three factors upon which the decision to stop sampling is based so that the search for the DOT MPT is approximated:

- how closely matched, in terms of frequency of occurrence, the translations in the sample set are,

- how many of the possible translations for the given input string are present in the set of sampled translations, and
- how certain we wish to be that the most frequent translation in the sample set is, in fact, the most probable translation according to the DOT model.

Alternative DOT ranking strategies

Of course, choosing the most probable translation as the best translation is not the only way to rank DOT translations. We can, for example, search instead for the **most probable representation** (that is, bilingual analysis) of the input string – i.e. the most probable linked source and target tree pair – and output the translation yielded by this representation as the best translation. In order to find this representation, we sample derivations according to their DOT sampling probabilities as described above. However, in this case our sample set comprises linked source and target tree pairs, rather than translations (as for DOT) or parses (as for DOP). We again use the BKS method to determine when enough samples have been taken, this time using the following stopping conditions:

- how closely matched, in terms of frequency of occurrence, the bilingual representations in the sample set are,
- how many of the possible bilingual representations for the given input string are present in the set of sampled representations, and
- how certain we wish to be that the most frequent bilingual representation in the sample set is, in fact, the most probable representation according to the model.

We can also search for the **most probable derivation** for DOT and, again, output the translation yielded by the target side of this representation. As for DOP, this is accomplished using the Viterbi algorithm, where only the fragment with the largest inside probability for each root node category pair is retained at each chart position.

Finally, we can search for the **shortest derivation** also – i.e. the derivation built using the fewest number of fragments – using the Viterbi algorithm by assigning all fragments equal probability $\frac{1}{p}$ (Bod, 2000e), generating the translation space which comprises all

shortest derivations and, where there is more than one, selecting the translation yielded by the most probable (according to the DOT model) of these shortest derivations.

5.3 Summary

We have examined the reasons for the disappointing performance of Poutsma (2000, 2003)'s DOT system and concluded that a more robust implementation is required to facilitate experiments using larger, more complex datasets. In order to achieve this, we have adapted the algorithms developed for Tree-DOP for use in our Tree-DOT system. In particular, we have outlined an efficient method for dynamically extracting the DOT fragment set from a bilingual treebank, an algorithm to generate the Tree-DOT translation space for a given input string, and four methods by which DOT output translations can be ranked. In addition, we have described a motivated way of pruning the DOT fragment set which takes into account the translational dependencies captured by DOT fragments. Thus, we have developed a system which will allow us to carry out a more intensive evaluation of the Tree-DOT model of translation than was possible heretofore. In the next chapter, we present the experiments we have carried out using this system and discuss in detail the results of our evaluation.

Chapter 6

Evaluating the DOT model

Previous experiments assessing the performance of the DOT model of translation were small in scale and the training data used was not ideally suited to the task (Poutsma, 1998, 2000, 2003). However, the limitations of the DOT implementation used to perform these experiments prevented a larger-scale, more informative assessment from being carried out. Having developed a DOT system which takes advantage of efficient algorithms developed for DOP, we are now in a position to perform translation experiments on a larger scale than heretofore. In this section, we describe our experiments in terms of the data used and the evaluation metrics upon which our assessment of the performance of the DOT model is based. We then go on to give translation accuracy results over variations in fragment depth and ranking strategies, and provide detailed analysis of our findings.

6.1 Experimental set-up

We present bidirectional Tree-DOT translation experiments on the English-French section of the HomeCentre corpus, which comprises 810 aligned translation pairs. Each sentence is annotated with an LFG representation comprising c-structure, ϕ-links and f-structure; we extracted only the c-structure (i.e. context-free phrase-structure tree) corresponding to each. As for the DOP parsing experiments presented in chapter 3, we preprocessed these trees by removing traces and empty categories and by removing unary-branching structures, i.e. substructures of the form $X \longrightarrow Y$ were replaced with the Y category. In addition, as it is frequently the case that alignments are not one-to-one, i.e. n source

sentences map to m target sentences, we combined groups of sentence representations forming a single translation unit into a single phrase-structure tree by simply inserting a root node PAIR such that each tree is a child of that pair. These changes were made fully automatically and, therefore, in a consistent manner; no manual alterations were made to the trees themselves. Manual intervention was necessary, however, in order to insert the translational links between paired trees; each English-French tree pair was linked only at the root node but DOT also requires links indicating translational equivalences at sub-structural level. This was done very simply by numbering each node in each tree and providing, for each tree pair, a list of node number pairs such that the presence of a number pair indicates a link between those two nodes.[1] Finally, our bilingual, sub-structurally aligned dataset was split randomly into 12 training/test splits such that all test words also appeared in the training set. Each of these splits comprises 80 test sentences and 720 training tree pairs; 6 of the splits have English as the source language and French as the target language and the other 6 splits have French as source and English as target.

We translate each test sentence[2] using the four ranking strategies – most probable translation, most probable parse,[3] most probable derivation and shortest derivation (referred to as MPT, MPP, MPD and SDer) – as they are described in section 5.2.4. Furthermore, we also prune the fragment base extracted from each training set with respect to link depth as defined in section 5.2.1, resulting in fragment bases comprising fragments of link depth 1, link depth 2 or less, link depth 3 or less and link depth 4 or less. As there are 16 ways of combining the ranking and pruning strategies, each test sentence in each split is translated in 16 different ways and the accuracy of the translations obtained is calculated over all splits for each combination.

It is not always the case that a translation can be produced for every test sentence

[1] Manual alignment is a time-consuming process which requires knowledge of both the source and target languages and is, consequently, not an ideal solution to the task of sub-structural alignment. An algorithm to accomplish this task automatically is described in (Groves et al., 2004). Reduced-scale, preliminary experiments on data aligned using this algorithm provide evidence that high-quality translations can also be produced using automatically-induced alignments; we discuss these findings in section 6.6.

[2] As for our Tree-DOP experiments, all experiments are carried out on a Pentium 4 with 2.39GHz CPU and 2Gb RAM.

[3] When computing both the most probable translation and the most probable parse for each input string using random sampling, we set the sampling thresholds P_{err} and θ described in section 2.5.1 to 0.01 and 2 respectively. We also set the maximum number of samples to 10,000 so that, in the event of there being two or more equally likely translations, sampling will terminate – this situation arose for 3 out of 480 sentences for English to French translation and for 4 out of 480 sentences for French to English translation.

in every split. Sentences which do not receive full translations fall into one of two categories: either there are fragments in the translation space which span all source words but no higher-level fragments to link them together, or there are source words to which no fragment corresponds.

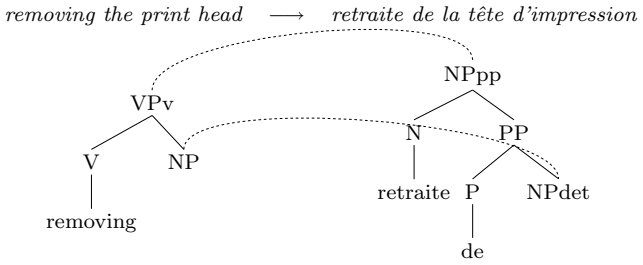
The first of these situations also arises in parsing but is more common for translation as source language fragments suffer from reduced compositionality due to the constraints imposed by their target language counterparts. (We revisit this issue in section 8.4.) We address this issue in precisely the same way for DOT as for DOP: we assign to the input sentence the best sequence of partial analyses according to the relevant ranking strategy. These sequences are combined by simply inserting fake root nodes of category DUMMY into the source and target trees such that the best source partial trees are siblings and the best target partial trees are siblings. As all source words are covered by fragments they all receive translations, meaning that the terminal symbols of the (partial) target tree comprise a string containing no untranslated words. This string is, however, not syntactically well-formed with respect to the training data.

The second situation – whereby there are gaps in the translation space such that there are source words to which no fragment corresponds – also arises as a result of the reduced compositionality of DOT fragments. In this situation, however, there are words in the source string for which, due to the context in which they appear, no translation can be generated. We do not attempt to translate these words by other means (such as a machine-readable dictionary), but rather leave these words untranslated in the output string. Again, we search for the most likely sequence of partial analyses according to each ranking strategy and combine them using the DUMMY category. However, each word for which we have no information is also tagged with the category DUMMY and this fake subtree is inserted into the target parse at the appropriate position. Thus, the output translation yielded by the target parse contains one or more source language words.

6.1.1 Translational divergence between English and French in the HomeCentre Corpus

The data on which we train and test the DOT model, the English-French HomeCentre corpus, comprises a Xerox printer manual; this manual was translated by professional translators and aligned and annotated at Xerox Parc. As one would expect, the translations it contains are of extremely high quality – in fact, we do not know which language was originally the source language for this dataset. As observed by Frank (1999), the corpus provides a rich source of both linguistic and translational complexity. While English and French are syntactically quite similar, they often differ significantly in the surface styles used to express the same concept. As we illustrate in the following discussion, translational divergences which generally prove challenging for MT models (Hutchins and Somers, 1992) are very much in evidence in this dataset.[4]

Instances of nominalisation are very frequent in the HomeCentre corpus. An example of a simple nominalisation is given in (6.1), where the English verb phrase *removing the print head* is realised as the noun phrase *retraite de la tête d'impression* in French.

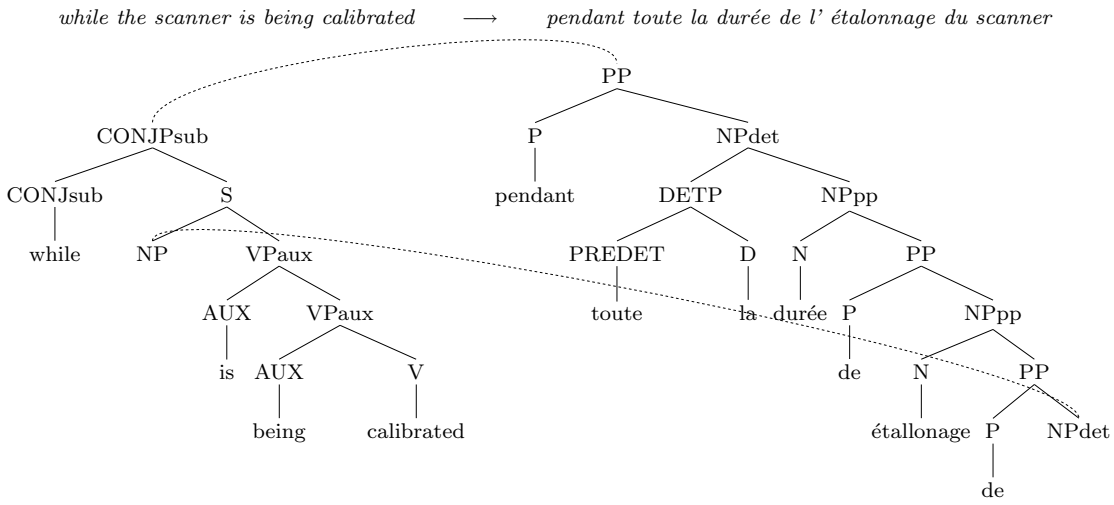


(6.1)

Instances of more complex nominalisations which incorporate further translational divergences are also common. Consider, for example, the translation pair given in (6.2). Firstly, we note the nominalisation: the English passive sentential form *the scanner is being calibrated* is realised as the French noun phrase *l'étalonnage du scanner*. However, we also observe the presence of relation-changing: the subject of this English sentential form, *the scanner*, functions as an oblique object in the French translation. In addition, this example

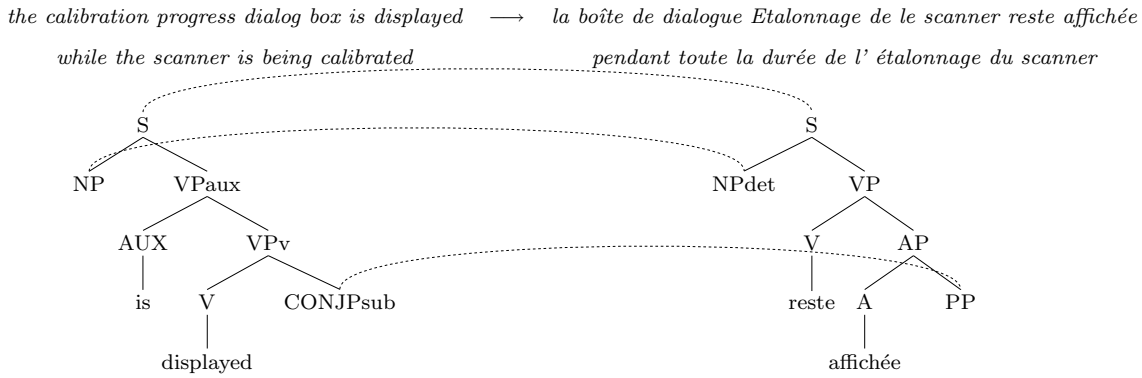
[4]For the sake of clarity, we focus this discussion on translation from English to French.

exhibits stylistic divergence, as *while* translates as *pendant toute la durée de*.



(6.2)

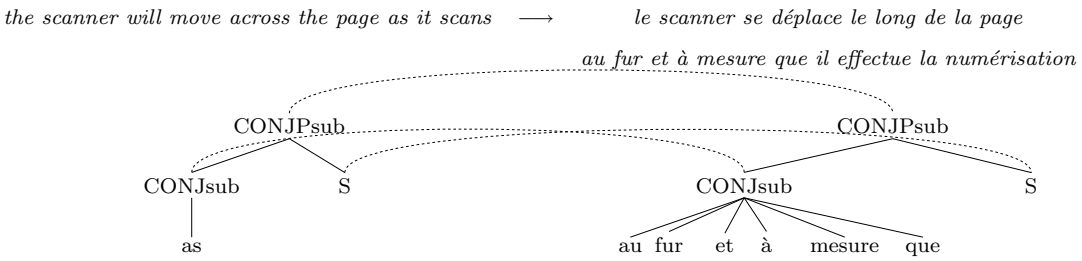
Another complex translation case which occurs in the HomeCentre corpus is that of head-switching, where the head word in the source language sentence translates as a non-head word in the target language realisation. An example of head-switching is given in (6.3). Here, the English verbal unit *is displayed* is realised in French as *reste affichée*; in this context, *reste* means (roughly) *remains* and *display* is realised as the advervbial modifier *affichée*. Thus, the head of the English sentence, the verb *display*, corresponds to the French non-head word *affichée*.



(6.3)

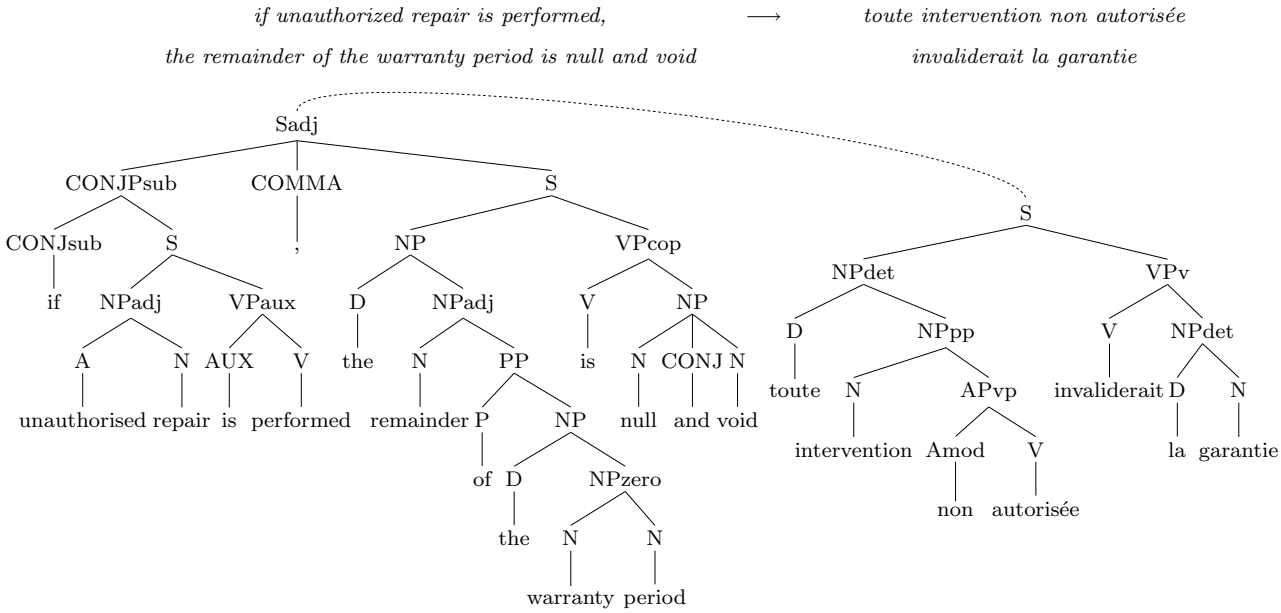
Of course, lexical divergences also occur frequently. In some instances, these divergences can be resolved in a straightforward manner. For example, we see in (6.4) that *as* in English can translate as *au fur et à mesure que* in French, but as the idiomatic reading of this French phrase is reflected in the parse assigned to the sentence, the overall shape

of the sentence can remain the same despite the complexity of the translation.



(6.4)

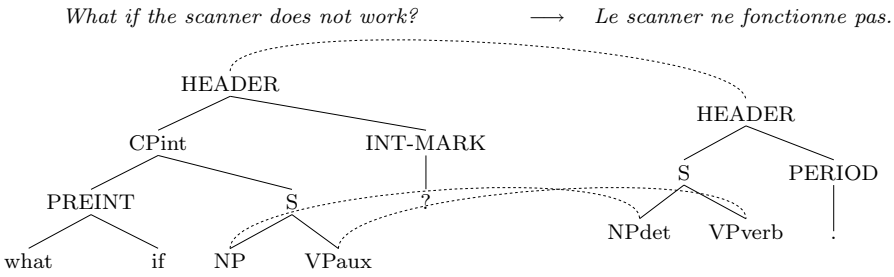
However, even for a relatively similar language pair, lexical divergence can cause source and target sentences expressing exactly the same concept to have completely different surface realisations. Consider, for example, the translation pair in (6.5). As there is no French phrase which is directly equivalent to the English expression *null and void*, the given French sentence *toute intervention non autorisée invaliderait la garantie* – which translates roughly as *any unauthorised action would invalidate the guarantee* – is entirely structurally dissimilar to its English counterpart.



(6.5)

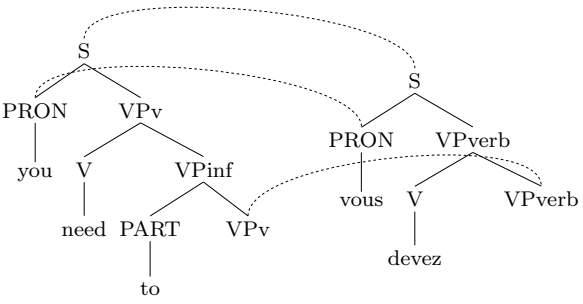
It is also common for sentences expressing exactly the same concept to have divergent surface realisations for purely stylistic reasons. For example, section headings in the English HomeCentre manual are often phrased as questions, whereas they generally appear in the declarative form in the French version. This is illustrated in example (6.6), where

the English section header *What if the scanner does not work?* corresponds to the French header *Le scanner ne fonctionne pas.*

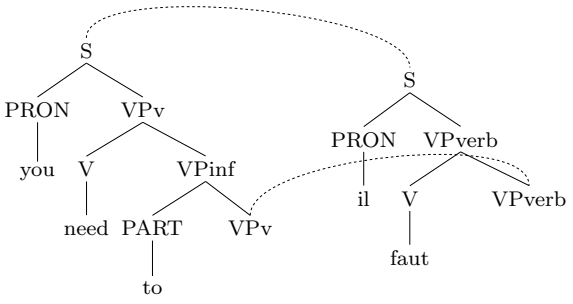


(6.6)

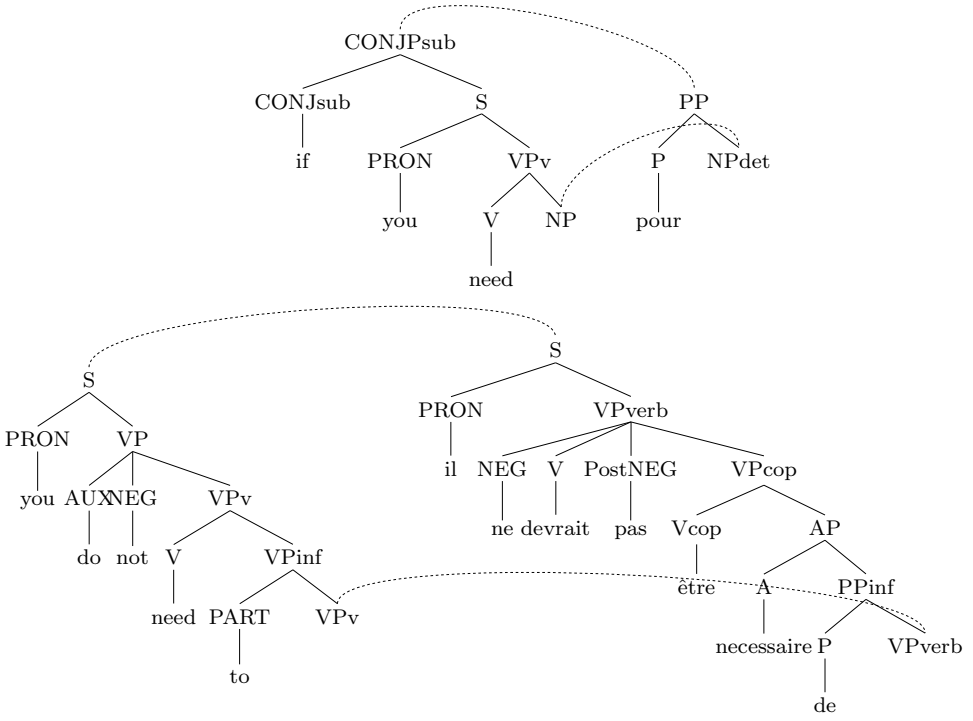
Finally, variation in how certain frequently-occurring words are translated, depending on the context in which the word appears, is also common. Examples (6.7) – (6.10) illustrate this phenomenon for the English verb *to need*. *you need to X* can be realised as both *vous devez X* and *il faut X* in French, as shown in examples (6.7) and (6.8). The realisation differs, however, where the object is nominal rather than sentential: *if you need X* is shown in (6.9) to translate as *pour X*. Finally, we show in example (6.10) that the negative *you do not need to X* can translate as *il ne devrait pas être necessaire de X*, which literally means *it should not be necessary to X* in English. We note that this is just a subset of the differing French realisations for the verb *to need* which occur in the HomeCentre corpus.



(6.7)



(6.8)



(6.9)

(6.10)

Thus, we conclude that the dataset we use to evaluate the DOT model contains many 'hard' translation examples, including cases of nominalisations, relation-changing, passivisation, headswitching and combinations thereof. Accordingly, the corpus would appear to present a challenge to any MT system. However, given that these cases are widespread in real data, most MT systems will be required to cope with such phenomena.

6.2 Evaluation metrics

Manual evaluation of MT output is informative but it is also time-consuming, expensive and not reusable. The advantages of automatic evaluation are obvious: it can be quick, cheap, language-independent, used for large-scale evaluation and, once developed, can be applied repeatedly to translation output during system development to assess changes made without incurring any extra costs. Crucially, however, automatic evaluation should also correlate highly with human judgements; consequently, developing and validating automatic MT evaluation metrics has proved challenging.

We describe four different automatic translation evaluation metrics: exact match, BLEU (Papineni et al., 2001, 2002), NIST (NIST, 2002; Doddington, 2002) and f-score (Melamed et al., 2003; Turian et al., 2003). These metrics all involve comparing output

translations (referred to as candidate translations) with their reference translations, but differ with respect to (i) how they measure the similarity between candidate and reference strings and (ii) how they reward the similarities and penalise the differences between those strings. A good automatic evaluation metric will be both sensitive and consistent, i.e. it will distinguish between systems of similar quality and will do so across varied reference translations (NIST, 2002).

NIST (2002) observe that automatic scoring is at its most reliable when reference translations are of high quality and the input sentences are from within the same genre. As our English and French data comprise Xerox printer manuals translated by professional translators, we feel that our experiments are particularly well suited to evaluation using automatic metrics. As we wish to highlight fluctuations in accuracy which result from (relatively) subtle changes to our system configuration – i.e. variation of fragment depth and ranking strategies – and none of the available metrics appears significantly better at reflecting such fluctuations than the others, we present accuracy scores corresponding to all four metrics[5] in order to give as clear a picture as possible as to how the DOT model performs.

6.2.1 The exact match metric

The exact match metric for measuring translation quality simply assigns score 1 to each translated sentence that exactly matches its corresponding reference translation and 0 otherwise. This is a very coarse-grained metric which can, nevertheless, yield useful information.

6.2.2 The BLEU metric

The BLEU metric (Papineni et al., 2001, 2002) evaluates MT system quality by comparing output translations to their reference translations in terms of the numbers of co-occurring n-grams. The main score calculated is the ***n*-gram precision** p_n for each pair of candidate and reference sentences. This score represents the proportion of n-word sequences in the

[5]We used version 11a of the BLEU/NIST evaluation software to calculate BLEU and NIST scores; we downloaded this software from http://www.nist.gov/speech/tests/mt/resources/scoring.htm. We calculated f-scores using GTM v1.2 downloaded from http://nlp.cs.nyu.edu/GTM/.

candidate translation which also occur in the reference translation. Importantly, if an n-gram occurs j times in the candidate translation and i times in the reference translation such that $i \leq j$ then this sequence is counted only i times; this corresponds to the intuition that "a reference word sequence should be considered exhausted after a matching candidate word sequence has been identified" (Papineni et al., 2001). Thus, n-gram precision p_n is calculated according to equation (6.11):

$$p_n = \frac{|c_n \cap r_n|}{|c_n|} \tag{6.11}$$

where

- c_n is the multiset of n-grams occurring in the candidate translation.
- r_n is the multiset of n-grams occurring in the reference translation.
- $|c_n|$ is the number of n-grams occurring in the candidate translation.
- $|c_n \cap r_n|$ is the number of n-grams occurring in c_n that also occur in r_n such that elements occurring j times in c_n and i times in r_n occur maximally i times in $|c_n \cap r_n|$.

As it is generally not the case that MT output is evaluated one sentence at a time, n-gram precision can also be calculated over sets of sentences. In this case, p_n is the proportion of co-occurring n-word sequences in the set over the total number of n-word sequences in the set.

While precision scores p_n can be obtained for any value of n,[6] Papineni et al. (2001) point out that greater robustness can be achieved by combining scores for all values of n into a single metric. It is not surprising that as the value of n increases, the score p_n decreases because longer matching word sequences are more difficult to find. If the average n-gram precision score is calculated without taking this factor into account (i.e. by simply summing the values for p_n and dividing by N, the largest value for n) then the scores for longer n-grams will be too small to have much influence on the final score. In order to make the BLEU metric more sensitive to longer n-grams, the combined score p_N is calculated by summing over the logarithm of each p_n multiplied by weight $1/N$ as given

[6] Scores can be obtained for any *reasonable* value of n; in (Papineni et al., 2001, 2002) the maximum value for n considered was 4.

in equation (6.12):

$$p_N = \exp(\sum_{n=1}^{N} \frac{1}{N} \log(p_n)) \tag{6.12}$$

A candidate translation which is longer than its reference translation is implicitly penalised during the calculation of p_n. In order to impose a corresponding penalty on candidate translations which are shorter than their reference translations, a *brevity penalty* BP is introduced and the combined precision score p_N is multiplied by this penalty. Papineni et al. (2001) state that BP is a decaying exponential in the length of the reference sentence over the length of the candidate sentence. This means that if the reference is the same length or longer than the candidate, then the penalty is 1, and greater than 1 if the candidate is shorter than the reference. Furthermore, if candidate c_x is 1 word shorter than its reference r_x and c_y is also 1 word shorter than r_y, but r_x is longer than r_y, then the BP for c_y should be greater than the BP for c_x. Thus, BP is calculated according to equation (6.13):

$$BP = e^{max(1-\frac{length(R)}{length(C)},0)} \tag{6.13}$$

Note that as calculating the brevity penalty for each sentence and averaging it over the set of sentences is considered by Papineni et al. (2001) to be unduly harsh, it is computed over the entire corpus, i.e. $length(R)$ is the number of words in the reference set and $length(C)$ the number of words in the candidate set. This penalty is then applied to the precision score for the entire candidate translation corpus according to equation (6.14):

$$BLEU = BP \cdot p_N \tag{6.14}$$

As the ranking behaviour is more visible in the log domain, Papineni et al. (2001) give equation (6.15):

$$\log BLEU = min(1 - \frac{length(R)}{length(C)}, 0) + \sum_{n=1}^{N} \frac{1}{N} \log(p_n) \tag{6.15}$$

In summary, the combination of n-gram precision and penalties for shorter translations mean that in order to achieve a high BLEU score a set of candidate translations must match the reference translation in length, in word choice and in word order (Papineni

et al., 2001).

6.2.3 The NIST metric

NIST (2002) investigated the sensitivity of the BLEU metric to systems whose output is of similar quality, and the consistency of BLEU as alternative sets of reference translations were provided. As a result of this investigation, they proposed a new score formulation – referred to simply as NIST – by making three changes to the BLEU measure.

The first issue addressed is that of n-gram informativeness: when calculating n-gram precision, BLEU assigns equal weight to each co-occurring n-gram according to formula (6.11), whereas NIST assigns more weight to co-occurring n-grams which are less frequent in the reference corpus. Effectively, this weighting works on the premise that finding an n-gram in both the candidate and reference translations which occurs frequently anyway gives less information about the quality of the MT output than finding a rare n-gram in both. Information weights are computed over n-gram counts for the set of reference translations according to formula (6.16):

$$Info(w_1...w_n) = \log_2(\frac{count(w_1...w_{n-1})}{count(w_1...w_n)}) \tag{6.16}$$

The NIST metric then factors these information weights into the BLEU n-gram precision formula given in (6.11) as shown in (6.17):

$$p_n = \frac{\sum_{w_1...w_n \in |c_n \cap r_n|} Info(w_1...w_n)}{|c_n|} \tag{6.17}$$

The second issue addressed concerns the combining of n-gram precision scores for each value of n into one score p_N. As given in equation (6.12), BLEU calculates the combined score p_N by summing over the logarithm of each p_n multiplied by weight $1/N$ in order to make the metric more sensitive to longer n-grams. However, (NIST, 2002) point out that this may be counterproductive; as this scoring is equally sensitive to variation in co-occurrence frequencies regardless of n's value, low co-occurrences for larger n-grams may result in unwarranted variation in scores. Thus, the NIST score combines p_n scores by simply taking their average according to formula (6.18) where n-gram sets C_n and R_n

are calculated over the full candidate and reference sets C and R:

$$p_N = \sum_{n=1}^{N} \frac{\sum_{w_1 \ldots w_n \in |C_n \cap R_n|} Info(w_1 \ldots w_n)}{|C_n|} \tag{6.18}$$

Finally, a change was also made to how the brevity penalty BP is calculated in order to minimise the impact on scores of small variations in translation length. This is done by introducing β, the value of which is chosen such that $BP = 0.5$ when the number of candidate words is $\frac{2}{3}$ the average number of reference translation words. Equation (6.19) gives the NIST formula for calculating BP:

$$BP = \exp(\beta \cdot \log_2(min(\frac{length(R)}{length(C)}, 1))) \tag{6.19}$$

As for BLEU, the overall NIST score is calculated by multiplying the combined n-gram precision score by the brevity penalty as given in (6.20):

$$NIST = BP \cdot p_N \tag{6.20}$$

A comparison of the performance of the BLEU and NIST metrics given in (NIST, 2002) indicates that NIST attains greater score stability and reliability for the corpora they studied.

6.2.4 The F-score metric

Melamed et al. (2003) and Turian et al. (2003) apply the standard measures of precision and recall to the evaluation of MT output. In general terms, precision and recall scores for candidate item C with respect to reference item R are calculated according to equations 6.21 and 6.22 respectively.

$$precision(C|R) = \frac{|C \cap R|}{|C|} \tag{6.21}$$

$$recall(C|R) = \frac{|C \cap R|}{|R|} \tag{6.22}$$

A method of calculating the intersection between two sentences must be defined in order to apply these methods to the evaluation of machine translated sentences against their

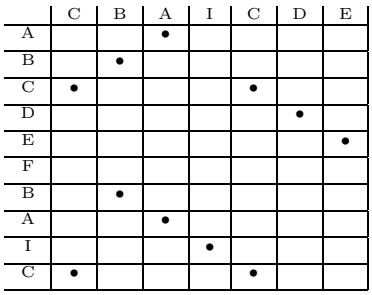


Figure 6.1: Bitext grid illustrating the relationship between an example candidate translation and its corresponding reference translation - the words of the candidate translation are shown from left to right across the top of the grid and the words of the reference translation are shown from top to bottom down the left-hand side of the grid. Each bullet, called a *hit*, indicates a word contained in both the candidate and reference strings. (This illustration is adapted from Figure 1 of (Melamed et al., 2003; Turian et al., 2003).)

reference translations. Precisely such a definition is given in (Melamed et al., 2003; Turian et al., 2003), where a bitext grid is used to show the intersection of two texts. An example of a bitext grid – adapted from Figure 1 of (Melamed et al., 2003) and (Turian et al., 2003) – is given in Figure 6.1, where the candidate string reads from left to right across the top of the grid and the reference string from top to bottom down the left-hand side of the grid. The intersections between these strings are marked by bullets (termed *hits*), i.e. each cell in the grid referring to the same candidate and reference word constitutes a point of intersection.

If we simply take $|C \cap R|$ to be the number of hits in the grid, then the count is over-estimated as some words will be counted more than once. For example, in Figure 6.1 the first candidate word 'C' gets two hits as this word appears twice in the reference translation – the total number of hits for each candidate word can be seen at a glance by simply counting the number of hits in its column in the grid. The concept of a *matching* (Melamed et al., 2003; Turian et al., 2003) is used to avoid this problem, where a matching is a reduced grid such that there is at most one hit in each row and each column. Examples of such matchings for the grid in Figure 6.1 are given in Figure 6.2. A *maximum matching* is a matching in which there are hits for as many candidate words as possible; in Figure 6.2, (b) and (c) are maximum matchings but (a) is not. The *maximum match size* (MMS)

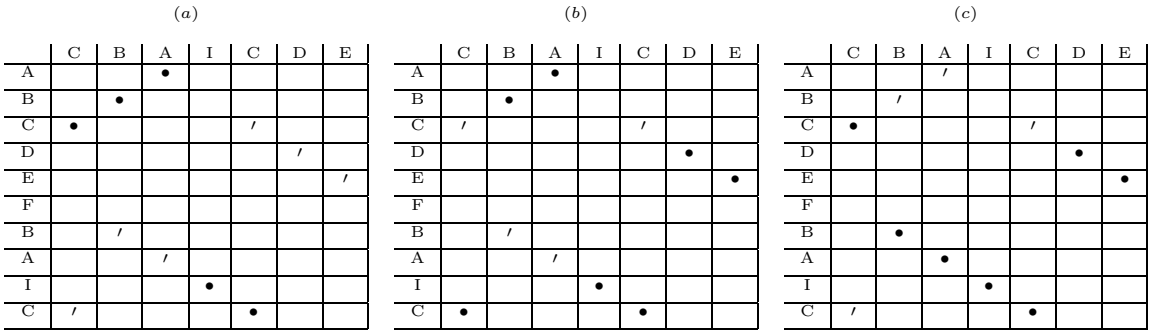


Figure 6.2: (a), (b) and (c) are examples of *matchings* for the grid in Figure 6.1. Hits which were in the original grid but are not contained in the matching are marked ′. In each matching, each row and column in the grid contains a single hit. (This illustration is adapted from Figure 1 of (Melamed et al., 2003; Turian et al., 2003).)

is the number of hits in a maximum matching – in Figure 6.2 (*b*) and (*c*), the MMS is 7 – and the MMS can never exceed the length of the shorter of the strings being compared.

The intersection between candidate and reference translations can be computed as the MMS (Melamed et al., 2003; Turian et al., 2003) and precision and recall calculated according to formulae (6.23) and (6.24).

$$precision(C|R) = \frac{MMS(C,R)}{|C|} \tag{6.23}$$

$$recall(C|R) = \frac{MMS(C,R)}{|R|} \tag{6.24}$$

However, these measurements do not penalise either for incorrect word order or non-contiguous hits, i.e. grids (*b*) and (*c*) in Figure 6.2 both contain the same number of hits and so receive exactly the same precision and recall scores despite the fact that grid (*c*) shows a matched 4-word sequence whereas the largest correct sequence shown in grid (*b*) has only 2 words. In order to reward correct word order, the definition of match size is generalised by treating *runs* as atomic units (Melamed et al., 2003; Turian et al., 2003). Each run is converted to an aligned block which is its minimum enclosing square; this is illustrated in Figure 6.3 (*b*) and (*c*) where the blocks of cells marked with circles correspond to the runs in Figure 6.2 (*b*) and (*c*).

The intersection of the reference sentence and candidate sentence can now be calculated

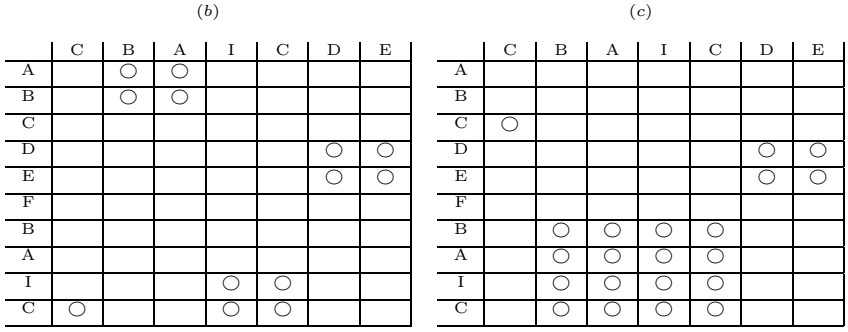


Figure 6.3: (b) and (c) are examples of maximum matchings for the grid in Figure 6.1. (This illustration is adapted from Figure 1 of (Melamed et al., 2003; Turian et al., 2003).)

in terms of the area of the aligned blocks by defining the weight of any single run as the square of its length. Thus, the calculation of match size MMS for a particular maximum matching M is calculated according to equation (6.25) (Melamed et al., 2003; Turian et al., 2003):

$$MMS(M) = \sqrt{\sum_{r \in M} length(r)^2} \tag{6.25}$$

According to this definition of match size, the grid in Figure 6.3 (b) is of size $\sqrt{1^2 + 2^2 + 2^2 + 2^2} = 3.61$ whereas grid (c) is of size $\sqrt{1^2 + 4^2 + 2^2} = 4.58$. As precision and recall are calculated according to equations 6.23 and 6.24 as before, grid (c) now scores higher than grid (b). As computing the MMS for any candidate and reference pair is NP-hard, Turian et al. (2003) use an approximation which finds the true maximum match size 99% of the time.

6.3 Results: English to French translation

In this section, we present the results of translation experiments carried out using our DOT system where translation took place from English into French. We consider the effects on accuracy of variation in the size of the fragment base and compare the performance of each of our ranking strategies MPT, MPP, MPD and SDer.

Table 6.1 shows, for each ranking strategy, the effect on translation accuracy of increasing the size of the fragments in the fragment base. We observe that the quality of the translations output increases steadily as fragment link depth increases for all ranking

	Most Probable Translation (MPT)			
	bleu	NIST	f-score	exact
1	0.4479	6.356	0.6712	30.21
2	0.5034	6.810	0.7035	37.92
3	0.5277	6.960	0.7179	40.00
4	**0.5343**	**7.037**	**0.7222**	**41.25**

	Most Probable Parse (MPP)			
	bleu	NIST	f-score	exact
1	0.4507	6.383	0.6733	30.62
2	0.4946	6.743	0.6990	37.50
3	0.5192	6.898	0.7135	38.96
4	**0.5216**	**6.928**	**0.7149**	**40.00**

	Most Probable Derivation (MPD)			
	bleu	NIST	f-score	exact
1	0.4572	6.439	0.6793	30.42
2	0.5069	6.856	0.7083	37.08
3	0.5269	6.979	0.7213	39.17
4	**0.5386**	**7.064**	**0.7257**	**41.04**

	Shortest Derivation (SDer)			
	bleu	NIST	f-score	exact
1	0.4168	6.105	0.6513	25.62
2	0.5080	6.851	0.7074	38.12
3	0.5314	6.994	0.7204	41.46
4	**0.5386**	**7.067**	**0.7254**	**42.29**

Table 6.1: Results for English to French DOT translation experiments which compare increases in link depth with translation accuracy for 4 metrics over all translations produced.

strategies and across all evaluation metrics.[7] Focussing in, for example, on exact match scores (which reflect the number of output translations exactly corresponding to their reference translations averaged over all splits), we see that for MPT ranking the overall increase in accuracy was 11.04%, for MPP ranking the overall increase was 9.38%, for MPD ranking the overall increase was 10.62%, and the greatest overall increase in exact match accuracy of 16.67% was achieved using SDer ranking. We also observe – again for all ranking strategies over all metrics – that the greatest increase in accuracy is achieved when going from link depth 1 fragments only to fragments of link depth 1 and link depth 2; the increases achieved when going from maximum link depth 2 to 3 and from link depth 3 to 4 are much more modest.

Table 6.2 shows, for each evaluation metric, how the different ranking strategies compare in terms of translation accuracy at each depth. At depth 1, we see that the Bleu, NIST and F-score metrics all show that best performance is achieved using MPD ranking whereas the exact match metric ranks MPP translations slightly ahead. At depth 2, the NIST and F-score metrics also show that MPD ranking performs best but the Bleu and exact match scores favour SDer ranking. At depth 3, Bleu, NIST and exact match all attribute best performance to MPD ranking but again the F-score measure places MPD ranking slightly ahead on accuracy. At depth 4, there is little to choose between MPD and SDer ranking according to Bleu, NIST and F-score but the exact match measure puts SDer ahead by 1.25%. Interestingly, ranking according to translation probability does not

[7] Exact match scores are given as percentages; the upper limit for bleu scores and f-scores is 1; NIST scores have no upper limit.

	Bleu Scores			
	1	2	3	4
MPT	0.4479	0.5034	0.5277	0.5343
MPP	0.4507	0.4946	0.5192	0.5216
MPD	**0.4572**	0.5069	0.5269	**0.5386**
SDer	0.4168	**0.5080**	**0.5314**	**0.5386**

	NIST Scores			
	1	2	3	4
MPT	6.356	6.810	6.960	7.037
MPP	6.383	6.743	6.898	6.928
MPD	**6.439**	**6.856**	6.979	7.064
SDer	6.105	6.851	**6.994**	**7.067**

	F-scores			
	1	2	3	4
MPT	0.6712	0.7035	0.7179	0.7222
MPP	0.6733	0.6990	0.7135	0.7149
MPD	**0.6793**	**0.7083**	**0.7213**	**0.7257**
SDer	0.6513	0.7074	0.7204	0.7254

	Exact Match Scores			
	1	2	3	4
MPT	30.21	37.92	40.00	41.25
MPP	**30.62**	37.50	38.96	40.00
MPD	30.42	37.08	39.17	41.04
SDer	25.62	**38.12**	**41.46**	**42.29**

Table 6.2: Results for English to French DOT translation experiments which compare ranking strategies over each link depth for each metric over all translations produced.

achieve highest accuracy at any depth according to any of the four evaluation measures. Focussing in on depth 4 – the depth at which all rankings give their best performance – we see that MPT output is consistently ranked in third place (behind MPD and SDer output) according to the Bleu, NIST and F-score metrics and takes second place over MPD ranking on the exact match metric by only 0.21%. Overall, these results show that the highest quality translations are generated using all fragments up to and including depth 4 and using either MPD or SDer ranking.

As discussed in section 6.1, not all translations produced are complete translations; some, while containing target-language words only, are not grammatical according to the training data as they were generated from sequences of partial representations whereas others also contain source-language words for which no translation was found in the training data. At depth 1, 65.62% of input sentences were assigned fully-formed, grammatical translations and 34.38% were assigned partial and/or ungrammatical translations. At depth 2, coverage increased slightly: 67.71% of sentences were assigned well-formed translations and 32.29% were assigned partial translations. No further improvements in coverage were observed at link depths 3 and 4. Tables 6.3 and 6.4 show accuracy evaluations for fully-formed, grammatical translations only (in the left-hand columns in each figure) and for partial and/or ungrammatical translations only (in the right-hand columns in each figure). In other words, the left-hand columns show results obtained by evaluating the (approximately) 55 sentences in each split which were fully translated against their reference translations and *excluding* the other reference translations from the reference set,

FULL TRANSLATIONS ONLY					PARTIAL TRANSLATIONS ONLY				
	Most Probable Translation (MPT)					Most Probable Translation (MPT)			
	bleu	NIST	f-score	exact		bleu	NIST	f-score	exact
1	0.6788	7.572	0.8217	45.40	1	0.2474	4.428	0.5300	**1.212**
2	0.7472	8.081	0.8580	56.00	2	0.2729	4.598	0.5444	0
3	0.7878	8.294	0.8797	59.08	3	0.2808	4.651	0.5503	0
4	**0.7906**	**8.331**	**0.8825**	**60.92**	4	**0.2913**	**4.735**	**0.5561**	0
	Most Probable Parse (MPP)					Most Probable Parse (MPP)			
	bleu	NIST	f-score	exact		bleu	NIST	f-score	exact
1	0.4863	6.625	0.7013	33.57	1	0.2124	3.710	0.4896	7.41
2	0.5321	6.992	0.7286	40.85	2	**0.2470**	**3.856**	0.5019	**11.11**
3	0.5608	7.177	0.7445	42.72	3	0.2414	3.850	**0.5080**	9.26
4	**0.5641**	**7.219**	**0.7477**	**43.66**	4	0.2410	3.800	0.5003	**11.11**
	Most Probable Derivation (MPD)					Most Probable Derivation (MPD)			
	bleu	NIST	f-score	exact		bleu	NIST	f-score	exact
1	0.6767	7.553	0.8216	45.40	1	0.2677	4.585	0.5464	**1.8180**
2	0.7480	8.082	0.8583	54.46	2	0.2787	4.667	0.5543	0.6452
3	0.7845	8.271	0.8793	57.54	3	0.2817	4.689	0.5575	0.6452
4	**0.7957**	**8.335**	**0.8841**	**60.31**	4	**0.2944**	**4.758**	**0.5614**	0.6452
	Shortest Derivation (SDer)					Shortest Derivation (SDer)			
	bleu	NIST	f-score	exact		bleu	NIST	f-score	exact
1	0.6312	7.260	0.7892	38.10	1	0.2304	4.278	0.5222	**1.8180**
2	0.7445	8.046	0.8520	56.00	2	0.2838	4.702	0.5577	0.6452
3	0.7902	8.313	0.8802	60.92	3	0.2859	4.685	0.5554	0.6452
4	**0.7992**	**8.364**	**0.8853**	**62.15**	4	**0.2910**	**4.745**	**0.5594**	0.6452

Table 6.3: Results for English to French DOT translation experiments which compare increases in link depth with translation accuracy for 4 metrics over partial and complete translations separately.

and the right-hand columns show corresponding results for the 25 or so sentences in each split which received translations yielded by partially-formed tree pairs.

Table 6.3 shows, for each ranking strategy, the effect on translation accuracy of increasing the size of the fragment base. We see that the results for complete translations confirm precisely what the results over all translations showed: translation accuracy improves steadily over all evaluation metrics for all ranking strategies as fragment depth increase and, again, the greatest increase in accuracy is always obtained when fragment depth increases from 1 to 2. On the other hand, the performance over partial translations is far less consistent. Firstly, we note that exact match results for partial translations do not offer much information as, realistically, we are surprised if *any* such translations exactly match their reference translations; we include exact match figures here for the sake of completeness only. Focussing on the other three metrics, we see that while accuracy again increases from depth 1 to depth 2, these increases are far smaller than for complete translations and there is little change in scores as depth increases to 3 and 4. Thus, we

FULL TRANSLATIONS ONLY

	Bleu Scores			
	1	2	3	4
MPT	**0.6788**	0.7472	0.7878	0.7906
MPP	0.4863	0.5321	0.5608	0.5641
MPD	0.6767	**0.7480**	0.7845	0.7957
SDer	0.6312	0.7445	**0.7902**	**0.7992**

	NIST Scores			
	1	2	3	4
MPT	**7.572**	8.081	8.294	8.331
MPP	6.625	6.992	7.177	7.219
MPD	7.553	**8.082**	8.271	8.335
SDer	7.260	8.046	**8.313**	**8.364**

	F-scores			
	1	2	3	4
MPT	**0.8217**	0.8580	0.8797	0.8825
MPP	0.7013	0.7286	0.7445	0.7477
MPD	0.8216	**0.8583**	0.8793	0.8841
SDer	0.7892	0.8520	**0.8802**	**0.8853**

	Exact Match Scores			
	1	2	3	4
MPT	**45.40**	**56.00**	59.08	60.92
MPP	33.57	40.85	42.72	43.66
MPD	**45.40**	54.46	57.54	60.31
SDer	38.10	**56.00**	**60.92**	**62.15**

PARTIAL TRANSLATIONS ONLY

	Bleu Scores			
	1	2	3	4
MPT	0.2474	0.2729	0.2808	0.2913
MPP	0.2124	0.2470	0.2414	0.2410
MPD	**0.2677**	0.2787	0.2817	**0.2944**
SDer	0.2304	**0.2838**	**0.2859**	0.2910

	NIST Scores			
	1	2	3	4
MPT	4.428	4.598	4.651	4.735
MPP	3.710	3.856	3.850	3.800
MPD	**4.585**	4.667	**4.689**	**4.758**
SDer	4.278	**4.702**	4.685	4.745

	F-scores			
	1	2	3	4
MPT	0.5300	0.5444	0.5503	0.5561
MPP	0.4896	0.5019	0.5080	0.5003
MPD	**0.5464**	0.5543	**0.5575**	**0.5614**
SDer	0.5222	**0.5577**	0.5554	0.5594

	Exact Match Scores			
	1	2	3	4
MPT	1.212	0	0	0
MPP	**7.407**	**11.110**	**9.2590**	**11.110**
MPD	1.818	0.6452	0.6452	0.6452
SDer	1.818	0.6452	0.6452	0.6452

Table 6.4: Results for English to French DOT translation experiments which compare ranking strategies over each link depth for each metric where partial and complete translations are evaluated separately.

conclude that increasing fragment depth does little to help in situations where sparse data is an issue.

Table 6.4 shows, for each evaluation metric, how the different ranking strategies compare in terms of translation accuracy at each depth. Here, the results for complete translations show that the SDer ranking method scores best according to all metrics at link depths 3 and 4; there was less difference between MPD and SDer rankings at these depths over all translations. Furthermore, we observe that MPT ranking scores better when evaluated over complete translations only than when all translations are evaluated together; it matches the performance of MPD ranking very closely at link depth 1 and there is little to choose between MPT, MPD and SDer rankings at link depths 2, 3 and 4. Over partial translations, on the other hand, according to Bleu, NIST and F-score, MPD gives the best overall performance at link depths 1 and 4 and SDer does best at link depth 2; at link depth 3, NIST and F-score favour MPD ranking whereas Bleu favours SDer ranking.

	Most Probable Translation (MPT)			
	bleu	NIST	f-score	exact
1	0.4990	6.745	0.7177	43.75
2	**0.5513**	**7.087**	**0.7463**	49.17
3	0.5447	7.040	0.7443	48.75
4	0.5494	7.075	**0.7463**	**49.38**

	Most Probable Parse (MPP)			
	bleu	NIST	f-score	exact
1	0.4915	6.659	0.7098	44.38
2	0.5406	7.010	0.7407	50.00
3	**0.5454**	7.024	0.7423	49.38
4	0.5449	**7.028**	**0.7427**	**50.21**

	Most Probable Derivation (MPD)			
	bleu	NIST	f-score	exact
1	0.4946	6.707	0.7119	44.79
2	0.5396	6.975	0.7376	49.38
3	**0.5436**	6.993	0.7386	49.79
4	0.5434	**7.013**	0.**7396**	**50.21**

	Shortest Derivation (SDer)			
	bleu	NIST	f-score	exact
1	0.4316	6.358	0.6832	36.46
2	0.5318	6.961	0.7343	48.54
3	0.5465	7.010	0.7401	50.00
4	**0.5488**	**7.044**	**0.7421**	**50.42**

Table 6.5: Results for French to English DOT translation experiments which compare increases in link depth with translation accuracy for 4 metrics over all translations produced.

6.4 Results: French to English translation

In this section, we present the results of translation experiments carried out using our DOT system where translation took place from French into English. As before, we consider the effects on accuracy of variation in the size of the fragment base and compare the performance of each of our ranking strategies MPT, MPP, MPD and SDer.

Table 6.5 shows, for each ranking strategy, the effect on translation accuracy of increasing the size of the fragments in the fragment base. Unlike the evidence presented for English to French translation, only the results for SDer ranking show consistent increases in translation accuracy as fragment depth increases. MPT ranking achieves its highest Bleu and NIST scores at link depth 2, equally high F-scores at link depths 2 and 4 and best exact match score at link depth 4. Furthermore, all four metrics show a decrease in accuracy as link depth 3 fragments are introduced, with an increase as link depth 4 fragments are introduced. Both MPP and MPD ranking achieves best NIST, F-score and exact match scores at link depth 4 and best Bleu score at link depth 3 but the differences between scores at link depths 3 and 4 are slight for both. Focussing in on exact match scores, we observe that the greatest overall increase in accuracy was again achieved using SDer ranking, as the link depth 4 score is 13.96% higher than the score at link depth 1. As before, the greatest increases in exact match accuracy are achieved when going from link depth 1 to link depth 2, and at link depths greater than 2 we see very modest increases at best.

Table 6.6 shows, for each evaluation metric, how the different ranking strategies com-

	Bleu Scores			
	1	2	3	4
MPT	**0.4990**	**0.5513**	0.5447	**0.5494**
MPP	0.4915	0.5406	0.5454	0.5449
MPD	0.4946	0.5396	0.5436	0.5434
SDer	0.4316	0.5318	**0.5465**	0.5488

	NIST Scores			
	1	2	3	4
MPT	**6.745**	**7.087**	**7.040**	**7.075**
MPP	6.659	7.010	7.024	7.028
MPD	6.707	6.975	6.993	7.013
SDer	6.358	6.961	7.010	7.044

	F-scores			
	1	2	3	4
MPT	**0.7177**	**0.7463**	**0.7443**	**0.7463**
MPP	0.7098	0.7407	0.7423	0.7427
MPD	0.7119	0.7376	0.7386	0.7396
SDer	0.6832	0.7343	0.7401	0.7421

	Exact Match Scores			
	1	2	3	4
MPT	43.75	49.17	48.75	49.38
MPP	44.38	**50.00**	49.38	50.21
MPD	**44.79**	49.38	49.79	50.21
SDer	36.46	48.54	**50.00**	**50.42**

Table 6.6: Results for French to English DOT translation experiments which compare ranking strategies over each link depth for each metric over all translations produced.

pare in terms of translation accuracy at each depth. The Bleu, NIST and F-score measures each indicate – with the exception of Bleu at link depth 3 – that the best performance at all depths is achieved by searching for the MPT. The exact match scores do not follow the same trends: MPD performs best at link depth 1, MPP at link depth 2 and SDer at link depths 3 and 4; the MPT is ranked third at link depths 1 and 2 and last at link depths 3 and 4. The evidence presented here does not allow us to conclude which combination of depth and ranking method gives the best result. According to the Bleu and NIST scores, best performance is at link depth 2 using MPT ranking. According to the F-scores, however, equally high accuracy is achieved using MPT ranking at link depths 2 and 4. Finally, according to the exact match scores, overall best performance is obtained using SDer ranking at fragment link depth 4.

Again, not all translations produced are complete translations; some, while containing target-language words only, are not grammatical according to the training data as they were generated from sequences of partial representations whereas others also contain source-language words for which no translation was found in the training data. At link depth 1, 69.79% of input sentences were assigned fully-formed, grammatical translations and 30.21% were assigned partial and/or ungrammatical translations. At link depth 2, coverage increased slightly: 72.71% of sentences were assigned well-formed translations and 27.29% were assigned partial translations. No further improvements in coverage were observed at link depths 3 and 4. The left-hand columns of Tables 6.7 and 6.8 show results obtained by evaluating the (approximately) 57 sentences in each split which were fully

FULL TRANSLATIONS ONLY					PARTIAL TRANSLATIONS ONLY				
	Most Probable Translation (MPT)					Most Probable Translation (MPT)			
	bleu	NIST	f-score	exact		bleu	NIST	f-score	exact
1	0.6370	7.468	0.8051	61.49	1	**0.2926**	**4.626**	**0.5938**	**2.759**
2	0.6988	7.935	0.8383	66.76	2	0.2923	4.541	**0.5938**	2.29
3	0.6964	7.922	0.8392	67.05	3	0.2792	4.460	0.5863	0
4	**0.7067**	**7.982**	**0.8438**	**67.91**	4	0.2742	4.435	0.5861	0
	Most Probable Parse (MPP)					Most Probable Parse (MPP)			
	bleu	NIST	f-score	exact		bleu	NIST	f-score	exact
1	0.5189	6.793	0.7266	47.07	1	0.3048	4.049	0.5334	**11.11**
2	0.5714	7.157	0.7613	53.15	2	0.3193	4.099	0.5241	**11.11**
3	0.5734	7.159	0.7620	52.48	3	**0.3382**	**4.192**	**0.5346**	**11.11**
4	**0.5791**	**7.190**	**0.7638**	**53.38**	4	0.3101	4.074	0.5264	**11.11**
	Most Probable Derivation (MPD)					Most Probable Derivation (MPD)			
	bleu	NIST	f-score	exact		bleu	NIST	f-score	exact
1	0.6405	7.505	0.8083	62.39	1	**0.2800**	**4.489**	**0.5754**	**4.138**
2	0.6903	7.899	0.8373	66.76	2	0.2775	4.366	0.5732	3.053
3	0.6998	7.930	0.8400	67.91	3	0.2727	4.350	0.5706	1.527
4	**0.7069**	**7.982**	**0.8455**	**68.48**	4	0.2611	4.294	0.5660	1.527
	Shortest Derivation (SDer)					Shortest Derivation (SDer)			
	bleu	NIST	f-score	exact		bleu	NIST	f-score	exact
1	0.5556	7.103	0.7717	50.45	1	0.2464	4.311	0.5589	**4.138**
2	0.6770	7.822	0.8313	65.62	2	**0.2796**	**4.392**	**0.5745**	3.053
3	0.7017	7.936	0.8407	68.19	3	0.2765	4.378	0.5729	1.527
4	**0.7104**	**7.999**	**0.8468**	**68.77**	4	0.2688	4.336	0.5705	1.527

Table 6.7: Results for French to English DOT translation experiments which compare increases in link depth with translation accuracy for 4 metrics over partial and complete translations separately.

translated against their reference translations and excluding the other reference translations from the reference set, and the right-hand columns show corresponding results for the 23 or so sentences in each split which received translations yielded by partially-formed tree pairs.

Table 6.7 shows, for each ranking strategy, the effect on translation accuracy of increasing the size of the fragments in the fragment base. Contrary to the equivalent evidence presented for all French to English translations and precisely in line with the evidence presented for English to French translation, the results for complete translations only show that best performance is achieved for all ranking strategies at link depth 4, although we do not always see an increase in accuracy as link depth increases from 2 to 3. The evidence over partial translations is again far less consistent. According to the Bleu and NIST measures, we see that performance for MPT and MPD ranking is at its best at link depth 1 and decreases steadily as depth increases. F-scores for these rankings show best performance at link depths 1 and 2 and similar performance at link depths 3 and 4 rather

FULL TRANSLATIONS ONLY

	Bleu Scores			
	1	2	3	4
MPT	0.6370	**0.6988**	0.6964	0.7067
MPP	0.5189	0.5714	0.5734	0.5791
MPD	**0.6405**	0.6903	0.6998	0.7069
SDer	0.5556	0.6770	**0.7017**	**0.7104**

	NIST Scores			
	1	2	3	4
MPT	7.468	**7.935**	7.922	7.982
MPP	6.793	7.157	7.159	7.190
MPD	**7.505**	7.899	7.930	7.982
SDer	7.103	7.822	**7.936**	**7.999**

	F-scores			
	1	2	3	4
MPT	0.8051	**0.8383**	0.8392	0.8438
MPP	0.7266	0.7613	0.7620	0.7638
MPD	**0.8083**	0.8373	0.8400	0.8455
SDer	0.7717	0.8313	**0.8407**	**0.8468**

	Exact Match Scores			
	1	2	3	4
MPT	61.49	**66.76**	67.05	67.91
MPP	47.07	53.15	52.48	53.38
MPD	**62.39**	**66.76**	67.91	68.48
SDer	50.45	65.62	**68.19**	**68.77**

PARTIAL TRANSLATIONS ONLY

	Bleu Scores			
	1	2	3	4
MPT	0.2926	0.2923	0.2792	0.2742
MPP	**0.3048**	**0.3193**	**0.3382**	**0.3101**
MPD	0.2800	0.2775	0.2727	0.2611
SDer	0.2464	0.2796	0.2765	0.2688

	NIST Scores			
	1	2	3	4
MPT	**4.626**	**4.541**	**4.460**	**4.435**
MPP	4.049	4.099	4.192	4.074
MPD	4.489	4.366	4.350	4.294
SDer	4.311	4.392	4.378	4.336

	F-scores			
	1	2	3	4
MPT	**0.5938**	**0.5938**	**0.5863**	**0.5861**
MPP	0.5334	0.5241	0.5346	0.5264
MPD	0.5754	0.5732	0.5706	0.5660
SDer	0.5589	0.5745	0.5729	0.5705

	Exact Match Scores			
	1	2	3	4
MPT	2.759	2.290	0	0
MPP	**11.11**	**11.11**	**11.11**	**11.11**
MPD	4.138	3.053	1.527	1.527
SDer	4.138	3.053	1.527	1.527

Table 6.8: Results for French to English DOT translation experiments which compare ranking strategies over each link depth for each metric where partial and complete translations are evaluated separately.

than consistent decreases. For MPP ranking, all three metrics show highest accuracy at link depth 3 but Bleu and NIST show small increases as link depth goes from 1 to 2 while F-score shows a decrease. The three measures all show the same trend for SDer ranking: highest accuracy is at link depth 2 and decreases are seen as fragment link depth goes from 2 to 3 and again from 3 to 4. Thus, contrary to the situation for English to French translation, the evidence *does* suggest that increasing fragment depth can help to improve output translation quality for some ranking methods in situations where sparse data is an issue.

Table 6.8 shows, for each evaluation metric, how the different ranking strategies compare in terms of translation accuracy at each depth. Here, the results for complete translations are inconsistent with the results achieved when all translations produced were evaluated. We saw in Table 6.6 that, over all translations, conflicting evidence from the four evaluation metrics meant that we could not come to a conclusion as to which combination of depth and ranking method gives the best result. These results do, however, show that

for Bleu, NIST and F-score, MPT ranking performs best (but at varying depths) whereas SDer at link depth 4 achieves the highest exact match score and MPT ranking does not achieve the best exact match score at any depth. In contrast, all four metrics show that overall best performance is achieved using SDer ranking and all fragments of depth 4 or less when complete translations are evaluated separately from partial translations. Furthermore, all four metrics show that SDer ranking also performs best at link depth 3, that MPD ranking performs best at link depth 1 and that MPT ranking performs best at link depth 2 (except that the exact match measure ranks MPT joint best with MPD ranking at link depth 2). When we look at the results of the evaluation over partial parses only we see that the conflicting evidence gathered when all translations are evaluated together was as a result of their influence. According to both NIST and F-score results, MPT ranking performs best at all link depths when only partial translations can be produced. While Bleu scores show that MPP ranking performs best at all link depths, it also ranks MPT ranking in second place (above MPD and SDer ranking) at all link depths. This performance over partial translations is clearly enough to tilt the overall balance in favour of MPT ranking *except* when performance is considered in terms of exact match accuracy. As partial translations are, in fact, invalid according to the DOT model, we consider that the results over complete translations only constitute a more accurate characterisation of the model. Accordingly, we conclude that, for French to English translation, best performance is achieved using SDer ranking at link depth 4.

6.5 Discussion

6.5.1 Does DOT improve over SMT on the HomeCentre corpus?

In order to provide a baseline against which we can assess the performance of the DOT model and, thus, put our results into context, we ran SMT experiments over the same training and test data.[8] For each split x we trained on the same training data train_x (but

[8]Training was carried out using Giza++ (Och and Ney, 2003) downloaded from http://www.fjoch.com/GIZA++.html. Translations were generated using the ISI ReWrite Decoder (Germann et al., 2001; Germann, 2003) downloaded from http://www.isi.edu/licensed-sw/rewrite-decoder/ and the CMU-Cambridge Statistical Language Modeling toolkit (Clarkson and Rosenfeld, 1997) downloaded from http://mi.eng.cam.ac.uk/ prc14/toolkit.html.

ENGLISH TO FRENCH TRANSLATION

	Bleu Score	NIST Score	F-score	Exact(%)
SMT	0.2686	4.984	0.6203	22.29
DOT (WORST)	0.4168	6.105	0.6513	25.62
DOT (BEST)	0.5386	7.067	0.7257	42.29

FRENCH TO ENGLISH TRANSLATION

	Bleu Score	NIST Score	F-score	Exact(%)
SMT	0.3076	5.819	0.6554	29.79
DOT (WORST)	0.4316	6.358	0.6832	36.46
DOT (BEST)	0.5494	7.075	0.7463	50.42

Table 6.9: Results for SMT translation experiments both from English to French and from French to English.

using sentence pairs rather than tree pairs) and then tested on the test data test_x and evaluated the output translations. We scored the SMT output exactly as before, using the exact match, Bleu, NIST and F-score metrics and averaging the scores over all splits.

We present the evaluation of SMT translation quality both when translating from English to French and from French to English in Table 6.9. For each metric, we compare the SMT score against both the best and worst DOT scores, regardless of which link depth and ranking strategy was used to achieve these best and worst scores. Even the worst DOT score (i.e. the combination of fragment depth and ranking method scoring lowest) for each metric scores better than the SMT system when trained and tested on the same data. Furthermore, the best DOT score for each metric scores significantly better than the SMT system. Examples where the DOT model generated better translations (relative to the reference translations) than the SMT model for the same input strings are given in Table 6.10. None of the SMT translations produced were of higher quality than the corresponding DOT translations, although translations of similar quality (again, with respect to the reference translations) were observed.

The SMT system translations score surprisingly well given that so little training data is available. This indicates that the data provided yields rich statistical information. However, comparison against the DOT translation scores shows that incorporating syntactic information, phrasal alignments and arbitrary translational dependencies results in a significant improvement in performance. Thus, we observe that the information elicited from equally small (but richly-annotated) training sets by the DOT model, combined with the

ENGLISH TO FRENCH TRANSLATION

Source	setting printer options
Reference	configuration de les options de impression
DOT	configuration de les options de impression
SMT	configuration options imprimante
Source	checking the status of your pending print jobs
Reference	vérification de l' état de les travaux en file d'attente de impression
DOT	vérification de l' état de les travaux de impression en attente
SMT	vérification de l' état de par attente travaux de impression
Source	what do the buttons and lights do ?
Reference	rôle de les boutons et de les voyants
DOT	rôle de les boutons et des voyants
SMT	courant procédez les boutons et voyants

FRENCH TO ENGLISH TRANSLATION

Source	modification de les options de impression enregistrées dans un fichier de préréglages
Reference	editing the printer options defined in a preset file
DOT	editing the printer options defined in a preset file
SMT	changing printer editing a preset file in options
Source	sélection de le HomeCentre comme imprimante par défaut
Reference	choosing the HomeCentre as your default printer
DOT	choosing the HomeCentre as your default printer
SMT	specifying HomeCentre as default choose the printer
Source	vous pouvez à présent imprimer une page de test pour vous assurer que l' imprimante fonctionne et pour vérifier la qualité d'impression .
Reference	you can print a test page to check the print quality and test the printer .
DOT	now you can print a test page to make sure the printer work and for check print quality .
SMT	you can work supported to you print a test page sure maintain printer and building to quality the print .

Table 6.10: Examples of differing translations produced by the DOT and SMT models for the same input string. (All DOT translations given here were produced by searching for the MPT using fragments of link depth 4 or less.)

DOT approach to generating and ranking translations, yields greater translation accuracy. Finally, as DOT makes use of available syntactic information whereas the SMT model used does not, we would like in the future to compare DOT performance to SMT systems which also exploit syntactic information, such as those described by Yamada and Knight (2001) and Charniak et al. (2003).

6.5.2 Do we improve on previous DOT experiments?

Meaningful comparison of the results published by Poutsma (2003) for the DOT model – described in section 5.1.2 – with the results presented here is difficult for a number of reasons. Firstly, bilingual treebanks of differing languages, text types and dimensions are used in each set of experiments. Secondly, Poutsma does not state how exactly he calculates the depth of each fragment so we do not know if he meant source subtree depth, target subtree depth or some middle ground. Thirdly, Poutsma's evaluation predates the development of the automatic evaluation metrics we have used to assess translation quality so his scoring methods (with the exception of the exact match metric) do not correspond to ours. Finally, much of Poutsma's discussion as to the quality of the translations produced by the DOT model centers around comparison with the translations output by the Systran machine translation system for the same sets of test sentences. We do not feel that such a comparison is appropriate for our experiments as our test data is domain-specific and uses specialised terminology, whereas Systran is a general purpose system.

The only possible comparison of results is on exact match accuracy over complete translations only produced using MPT ranking. Poutsma's results show that exact match figures for experiments from English to German are 16%–19% and 13%–15% from German to English. As he states that he assigns sentences for which no translation was produced to a *none* category (Poutsma, 2003):349 but the published tables of results do not indicate how many sentences fell into this category (Poutsma, 2003):350, we assume that the scores given are based on a distribution from which untranslated sentences were excluded. Consequently, the corresponding scores in our evaluation are those calculated over complete translations only. Our results are significantly better, showing that the appropriate exact match figures for experiments from English to French are 45.50%–60.90%

ALL ENGLISH TRANSLATIONS					ALL FRENCH TRANSLATIONS				
	CPU seconds/sentence					CPU seconds/sentence			
	MPT	MPP	MPD	SDer		MPT	MPP	MPD	SDer
1	1.39	1.33	0.29	0.30	1	0.72	3.73	3.12	3.13
2	2.06	1.55	0.57	0.58	2	1.16	3.85	3.53	3.58
3	3.05	2.28	1.40	1.41	3	2.32	4.96	4.62	4.64
4	12.8	11.9	11.3	11.1	4	18.9	21.5	21.1	20.8

Table 6.11: Average time taken to translate each sentence for all link depths and ranking strategies, and both translation directions.

and 61.49%–67.91% from French to English.

6.5.3 Ranking algorithms: efficiency vs. accuracy

As discussed in section 3.2.5, algorithms such as the Monte Carlo method which approximate an NP-hard search problem are generally not adopted if a deterministic alternative can be found which does not introduce an unacceptable degradation in performance. We have presented experiments where four ranking strategies are compared; two of these strategies – MPT and MPP – use random sampling to approximate the search space whereas the other two – MPD and SDer – use the Viterbi algorithm. In this section, we look at how these ranking strategies compare in terms of efficiency.

Table 6.11 gives the average number of seconds required to translate each sentence at each depth and using each of the four ranking strategies – times for English to French translation are given on the left and times for French to English translation are given on the right. The times given represent full processing time for each sentence, i.e. the time taken to apply the two-phase analysis algorithm in order to determine the translation space and the time taken to select the best output translation.[9] Table 6.12 gives, for MPT and MPP ranking, the average number of samples taken when disambiguating at each fragment depth – again the left table refers to English as source language and the right to French as source language.

Focussing firstly on the average times over all translations given in Table 6.11, we see that – not surprisingly – the time taken to translate each sentence increases as fragment link depth increases, with a large increase from link depth 3 to link depth 4. The extra time

[9] We do not give separate disambiguation times as parsing and disambiguation are inextricably linked for the MPD and SDer ranking methods.

ALL ENGLISH TRANSLATIONS

	samples/sentence	
	MPT	MPP
1	206.3	273.1
2	166.9	260.2
3	134.3	214.2
4	106.7	197.9

ALL FRENCH TRANSLATIONS

	samples/sentence	
	MPT	MPP
1	56.1	160.9
2	39.2	95.7
3	34.6	91.6
4	30.3	93.1

Table 6.12: Average number of samples taken when selecting an output translation for MPT and MPP ranking, for all link depths and both translation directions.

taken for each sentence at greater depths is spent building the translation space (which contains increasing numbers of fragments) rather than ranking the output translations. In fact, we see from Table 6.12 that fewer samples are taken per sentence for both MPT and MPP ranking as fragment depth increases; as seen for parsing in section 3.2.5, as more contextual information is introduced it becomes easier to determine which translation (for MPT) and representation (for MPP) is most probable.

Again looking at Table 6.11 but this time comparing ranking algorithms, we observe that for English to French translation at each depth, MPT ranking takes longest, followed by MPP ranking and MPD, and SDer rankings are fastest but the difference between the fastest and slowest at link depth 4 is just 1.7 seconds. The opposite, however, holds for French to English translation: MPP ranking is slowest, followed by MPD and SDer, and MPT ranking is consistently fastest. (Again, the difference in time taken between fastest and slowest at link depth 4 is small.) This allows us to conclude that not only do the ranking methods which require random sampling *not* take significantly longer to process each sentence than our ranking strategies based on the Viterbi algorithm, but in some instances they actually arrive at a solution more quickly. In order to investigate why this is the case, we look more closely at the differences in times taken by the MPT and MPP ranking methods.

For English to French translation, calculating the MPT takes longer than calculating the MPP at each depth whereas, for French to English translation, the opposite holds. There are two issues at play here. Firstly, we see from Table 6.12 that more samples per sentence are required to determine the MPP than the MPT. It is sometimes the case that although more than one pair of analyses can be generated for a given input string, all of those analyses yield the same target translation. In this situation, sampling is

required to determine the most probable analysis but not to determine the most probable translation. Clearly, this has an effect on the average numbers of samples taken for each strategy and, consequently, the average sentence processing times. However, while this explains why, for French to English translation, searching for the MPT is faster, it does not explain why the opposite holds for English to French translation. Recall that one of the factors upon which the decision to stop sampling is based is the total number of items that are being ranked, i.e. when ranking according to translation probability, we consider the total number of translations possible, and when ranking according to (paired) analysis probability we consider the total number of analyses possible. However, while the total number of analyses possible can be computed efficiently during construction of the translation space, computing the total number of translations possible is more costly as the combination of terminal symbols and frontier substitution sites yielded by each target language fragment in the translation space must be considered. Whether or not the time invested in determining exactly how many translations there are for each string is time well spent depends on whether ranking according to translation probability gives higher quality output than ranking according to analysis probability. Looking back to the results for complete translations only given in Tables 6.3 and 6.7, we see that MPT ranking significantly outperforms MPP ranking for both translation directions at all depths and for all metrics. Thus, we conclude that, if choosing between MPT and MPP ranking, determining the number of possible translations for each string *is* time well spent.

In sections 6.3 and 6.4 we looked at the accuracy achieved when complete translations (i.e. translations which, according to the training data, are fully grammatical) are evaluated separately from partial translations (i.e. translations which, according to the training data, are incomplete and/or ungrammatical), and found that not distinguishing between complete and partial translations distorts the evidence as to which system configuration gives higher translation accuracy. We now turn our attention to the differences in efficiency over complete and partial translations. Table 6.13 gives the average seconds per sentence where full translations were generated and the average seconds per sentence where only partial translations were generated for English as source language (in the left column) and French as source language (in the right column). Table 6.14 gives the average

FULL ENGLISH TRANSLATIONS

	CPU seconds/sentence			
	MPT	MPP	MPD	SDer
1	0.73	1.09	0.19	0.18
2	1.63	1.10	0.38	0.41
3	2.44	1.81	0.99	1.00
4	9.10	11.5	7.59	7.51

FULL FRENCH TRANSLATIONS

	CPU seconds/sentence			
	MPT	MPP	MPD	SDer
1	0.44	3.76	1.49	1.45
2	0.83	3.94	1.78	1.81
3	1.83	4.97	2.69	2.67
4	15.5	21.5	16.2	16.0

PARTIAL ENGLISH TRANSLATIONS

	CPU seconds/sentence			
	MPT	MPP	MPD	SDer
1	2.65	3.26	0.47	0.52
2	2.96	5.15	0.97	0.94
3	4.32	6.02	2.27	2.25
4	20.5	15.4	19.0	18.7

PARTIAL FRENCH TRANSLATIONS

	CPU seconds/sentence			
	MPT	MPP	MPD	SDer
1	1.38	3.36	6.88	7.03
2	2.02	2.64	8.18	8.31
3	3.60	4.83	9.75	9.86
4	27.8	21.2	34.2	33.8

Table 6.13: Average time taken to translate each sentence for all link depths and ranking strategies, and both translation directions, where we distinguish between complete and partial translations.

FULL ENGLISH TRANSLATIONS

	samples/sentence	
	MPT	MPP
1	67.7	226.9
2	44.3	170.2
3	25.3	113.2
4	21.5	106.3

FULL FRENCH TRANSLATIONS

	samples/sentence	
	MPT	MPP
1	28.2	138.0
2	18.4	82.9
3	13.8	67.5
4	13.4	69.4

PARTIAL ENGLISH TRANSLATIONS

	samples/sentence	
	MPT	MPP
1	471.0	637.4
2	424.1	970.7
3	362.9	1011.0
4	285.3	920.6

PARTIAL FRENCH TRANSLATIONS

	samples/sentence	
	MPT	MPP
1	120.5	442.9
2	94.8	254.1
3	90.0	388.1
4	75.4	385.6

Table 6.14: Average number of samples taken when selecting an output translation for MPT and MPP ranking, for all link depths and both translation directions, where we distinguish between complete and partial translations.

samples per sentence (for MPT and MPP ranking) where full translations were generated and, separately, where partial translations were generated; again, results for English as source language are given in the left column and for French as source language in the right column.

Table 6.13 indicates clearly that sentences for which only partial translations can be generated take significantly longer to process at all depths than those for which full translations can be generated for both translation directions; the exception to this is MPP ranking for French to English translation where we see that times are similar for both. However, as was the case over all translations, processing times for MPT ranking over com-

plete translations are slightly higher than MPD and SDer ranking for English as source language and slightly lower for French as source language. Although the times taken are significantly longer, similar trends are observed for the processing of sentences for which only partial translations are generated. We see from Table 6.14 that while, as before, sample sizes decrease as fragment depth increases, significantly more samples are taken when generating partial translations; this is in accordance with the increased time taken to process these translations.

Given that the time taken to generate partial translations is longer than to generate complete translations, that the quality of those translations is significantly poorer and that they are, in any case, assigned zero probability by the DOT model, whether or not we should attempt to generate such translations at all is debatable. It is important to point out, however, that much of the time required to process these sentences is spent in *determining* that they cannot be translated grammatically by the model, so not outputting such translations may not save us a great deal of time. Furthermore, we note that while partial translations are not of as high quality as complete translations, they still receive reasonable evaluation scores. Thus, we conclude that generating translations which are ill-formed with respect to the model and training data is worthwhile.

We looked in detail in sections 6.3 and 6.4 at the accuracy of each system configuration and concluded that, overall, the highest quality translations were produced by searching for the shortest derivation and using all fragments of link depth 4 or less. Having also considered the efficiency of each configuration and observed that for the configurations which give the best accuracy (MPT and SDer at link depth 4), there is little difference in efficiency – MPT takes, on average, 1.7 seconds per sentence longer than SDer when translating from English to French but SDer takes 1.9 seconds per sentence longer when translating from French to English. Thus, for the DOT model over the HomeCentre corpus, we conclude that there is no need to sacrifice accuracy for efficiency as the most accurate model – SDer at link depth 4 – is as efficient as its closest competitor.

FULL ENGLISH TRANSLATIONS					FULL FRENCH TRANSLATIONS				
	Exact Match Scores					Exact Match Scores			
	1	2	3	4		1	2	3	4
MPT	**45.40**	**56.00**	59.08	60.92	MPT	61.49	**66.76**	67.05	67.91
MPP	33.57	40.85	42.72	43.66	MPP	47.07	53.15	52.48	53.38
MPD	**45.40**	54.46	57.54	60.31	MPD	**62.39**	**66.76**	67.91	68.48
SDer	38.10	**56.00**	**60.92**	**62.15**	SDer	50.45	65.62	**68.19**	**68.77**

Table 6.15: Exact match accuracy for complete translations only.

6.5.4 How come MPP ranking performs so poorly?

The results presented in Tables 6.4 and 6.8, which illustrate translation accuracy over complete translations only, show that MPP ranking gives significantly lower quality translations than the other three ranking methods. For example, if we focus in on the exact match scores, which we have repeated in Table 6.15 for the sake of convenience, we see that for English as source language the MPP score is, at best, 11.83% lower than the best score and, at worst, 18.49% lower than the best score. The situation for French as source language is similar: the MPP score is, at best, 13.61% lower than the best score and, at worst, 15.71% lower.

In order to ascertain why this is the case, we look more closely at the ranking imposed by searching for the MPP. Recall that this method ranks representations rather than translations, where each representation comprises a source-language parse tree along with a target-language parse tree, the terminals of which constitute the output translation. The MPD and SDer methods also rank representations rather than translations, MPD according to derivation probability and SDer according to derivation length (backing off to derivation probability). Thus, we can also look at the quality of the parse tree assigned to each input string by each of these three methods in order to determine analysis accuracy.[10] Accordingly, in Table 6.16 we present an evaluation of the source-language parse trees assigned to each input string which received a full translation where the F-score and exact match scores were calculated as described in section 3.2.2. These results show that the quality of the source-language parses selected by each method is very similar at each depth – the biggest divergence in accuracy is for French parses on the exact match metric at link depth 3 where the MPP score is 5.26% lower than the best score. Furthermore, overall

[10]Note that, as the MPT method ranks strings rather than representations and these strings may have been yielded by more than one representation, this method does not select any single representation as the best representation for the input string. Thus, we cannot evaluate parse accuracy for MPT ranking.

FULL ENGLISH TRANSLATIONS

	F-Score		
	MPP	MPD	SDer
1	**98.17**	97.65	97.13
2	98.69	98.56	**98.95**
3	**98.95**	98.56	98.56
4	**98.95**	**98.95**	98.56

	Exact Match		
	MPP	MPD	SDer
1	**90.38**	88.46	86.54
2	94.23	94.23	**96.15**
3	**96.15**	94.23	94.23
4	**96.15**	**96.15**	94.23

FULL FRENCH TRANSLATIONS

	F-Score		
	MPP	MPD	SDer
1	93.96	93.96	**94.72**
2	**95.92**	95.58	95.41
3	95.75	95.92	**96.09**
4	96.43	96.43	**96.60**

	Exact Match		
	MPP	MPD	SDer
1	67.92	67.92	**73.58**
2	**78.95**	77.19	75.44
3	75.44	78.95	**80.70**
4	84.21	84.21	**85.96**

Table 6.16: Exact match accuracy for complete translations only.

parse quality for each of these methods is extremely high.

We have observed that the number of samples decreases at each link depth for MPP ranking, meaning that it becomes easier to discern which representation is more likely as depth increases. We have observed that MPP parse accuracy either increases or stays the same as depth increases and we have observed that MPP parse accuracy is similar to that of MPD and SDer ranking. Thus, the only explanation for poor translation performance using this method seems to be that ranking translations for the input string according to representation probability is simply not the most appropriate way of selecting the best translation.

6.5.5 Does the DOP Hypothesis also apply to DOT?

The DOP Hypothesis states that parse accuracy improves as larger fragments are included in the fragment base. As discussed in section 3.2.5, our parsing experiments on the English and French sections of the HomeCentre corpus do not uphold the DOP Hypothesis due to the bias in the parameter estimation method which means that, although it becomes easier to tell which parse is best as fragment depth increases (as evidenced by sample sizes), parse accuracy deteriorates. Thus, in section 3.2.5 we concluded that we were, indeed, establishing correctly which parse was most probable according to the DOP model, but the bias in favour of larger fragments meant that the most probable parse corresponded to the best parse less frequently as fragment size increased.

In order to illustrate the problem fully, we presented information on the size of the fragment base at each link depth, showing how the proportion of fragment space occupied

ENGLISH TO FRENCH FRAGMENT SETS

	depth 1	depth 2	depth 3	depth 4
Fragments per training set:	6,140	29,081	148,165	1,956,786
CPU seconds to compile fragment set:†	30.8	34.0	37.8	263.2

FRENCH TO ENGLISH FRAGMENT SETS

	depth 1	depth 2	depth 3	depth 4
Fragments per training set:	6,197	29,355	150,460	2,012,632
CPU seconds to compile fragment set:†	31.3	33.5	38.8	690.0

†We have included the information on the time taken to compile each fragment set in order to (i) illustrate the relative increase in time required to compile increasingly larger fragment sets and (ii) be as comprehensive as possible in documenting our experiments. However, the absolute values are not informative as this module has not been optimised for speed.

Table 6.17: Comparison of training set details in terms of the number of fragments at each link depth and the time taken to compile each fragment set at each link depth.

ENGLISH TO FRENCH

	%(d=1)	%(d=2)	%(d=3)	%(d=4)
$d \leq 1$	100	-	-	-
$d \leq 2$	17.43	82.57	-	-
$d \leq 3$	3.35	15.86	80.79	-
$d \leq 4$	0.29	1.36	6.92	91.43

FRENCH TO ENGLISH

	%(d=1)	%(d=2)	%(d=3)	%(d=4)
$d \leq 1$	100	-	-	-
$d \leq 2$	20.62	79.38	-	-
$d \leq 3$	3.33	15.78	80.89	-
$d \leq 4$	0.28	1.34	6.84	91.54

Table 6.18: Comparison of the proportion of the fragment set occupied by each fragment depth (d) as overall fragment depth increases.

by the maximum fragment link depth increased as link depth increased. In Tables 6.17 and 6.18, we present the corresponding information for the bilingual DOT fragment sets. These tables show that the numbers of fragments and the distributions they occupy are very similar to those shown for parsing. As we also establish fragment probabilities in the same way as for parsing (i.e. by calculating their relative frequencies), we expect to see the same deterioration in output quality as we did for our parsing experiments.

Contrary to expectations, however, our evaluation of translation accuracy does actually show that the DOP Hypothesis, for the most part, holds for DOT. The output translations given in Table 6.19 illustrate how translation quality can differ depending on the dimensions of the fragment base. For English to French translation, evaluations over all translations and over complete translations only and for all ranking algorithms and metrics show that translation accuracy increases as fragment size increases. For French to English translation, the results are not quite so consistent: evaluations over complete translations show that translation accuracy improves as depth increases for all ranking

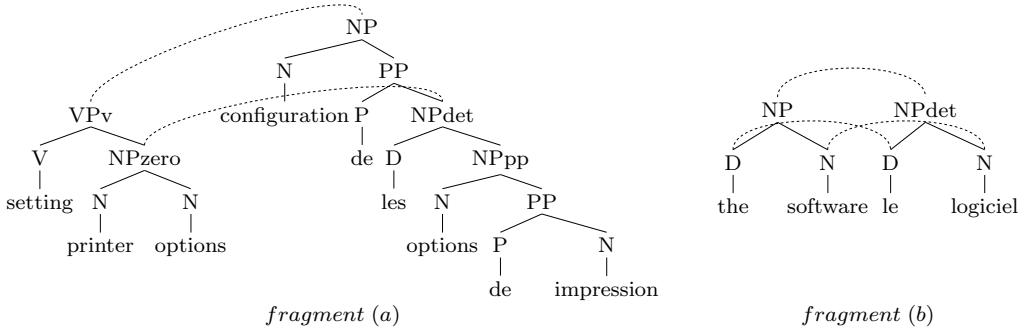


Figure 6.4: $link\_depth(a) = 2$, $link\_depth(b) = 2$

algorithms and metrics translation except for MPT ranking at link depth 3. Evaluation over all translations produced show consistent increases for SDer ranking over all metrics, consistent increases for MPD and MPP ranking over all metrics except Bleu and consistent increases for MPT ranking only using the exact match metric.

We believe that the observed improvements in translation accuracy as fragment depth increases – which we have achieved *despite* the known bias in our probability estimation method – are due to the combination of the constraints imposed on the fragment set by the presence of translational links and how we calculate the depth of bilingual fragments. Firstly, as discussed in section 5.2.2, the source-language subtrees and target-language subtrees extracted according to the DOT fragmentation definition are subsets of the source-language subtrees and target-language subtrees extracted according to the DOP fragmentation definition. This is because the translational links themselves act as a pruning method on the set of DOT fragments extracted by effectively specifying that only some of the nodes in each tree can be designated root nodes and substitution sites. Thus, from a DOP perspective, we do *not* use all possible fragments at each depth. Secondly, as fragment depth is defined in terms of *link depth* as described in section 5.2.1, the notion of depth for DOT fragments is not the same as for DOP fragments. As illustrated in Figure 6.4 – where fragments (a) and (b) have the same link depths despite being of very different dimensions – the actual size of the DOT subtrees comprising fragments of the same link depth can be radically different and, consequently, the amount of contextual information captured by each fragment can also be radically different. This is in marked contrast to the DOP situation, where controlling fragment depth essentially means exerting strict control over the amount of contextual information which can be expressed by each fragment.

ENGLISH TO FRENCH TRANSLATION

Source	in the print dialog box [see next page] , choose the options you want and click OK
Reference	dans la boîte de dialogue Impression [voir page suivante] , sélectionnez les options voulues et cliquez sur OK .
link depth 1	lorsque les boîte de dialogue de Imprimer se affiche [déterminer suivante page] , choisissez les options voulues et cliquez sur OK ;
link depth 2	lorsque les boîte de dialogue de Imprimer se affiche [déterminer suivante page] , choisissez les options voulues et cliquez sur OK ;
link depth 3	lorsque la boîte de dialogue Impression se affiche [voir page suivante] , sélectionner les options voulues et cliquez sur OK
link depth 4	dans la boîte de dialogue Impression [voir page suivante] , sélectionner les options voulues et cliquez sur OK
Source	on the Pagis tool bar , click Copy .
Reference	sur la barre d'outils de Pagis , cliquez sur Copier .
link depth 1	sur le barre d'outils de Pagis , cliquez sur Copier .
link depth 2	sur le barre d'outils de Pagis , cliquez sur Copier .
link depth 3	dans la barre d'outils de Pagis , cliquez sur Copier .
link depth 4	sur la barre d'outils de Pagis , cliquez sur Copier .

FRENCH TO ENGLISH TRANSLATION

Source	débranchez le cordon d'alimentation de la prise murale .
Reference	unplug the power cord from the wall outlet .
link depth 1	disconnect your power supply of the wall socket .
link depth 2	disconnect the power cord of the wall outlet .
link depth 3	disconnect the power cord from the wall socket .
link depth 4	disconnect the power cord from the wall outlet .
Source	dans la boîte de dialogue Impression , cliquez sur Propriétés .
Reference	in the print dialog box , click Properties [or Setup] .
link depth 1	to the print dialog box click properties [or Setup] .
link depth 2	in the print dialog box click properties [or Setup] .
link depth 3	in the print dialog box , click properties [or Setup] .
link depth 4	in the print dialog box , click properties [or Setup] .

Table 6.19: Examples showing how DOT output translations can differ depending on the dimensions of the fragment base.

We suggest, therefore, that the improved parameter estimation methods developed for DOP and described in section 2.6 should also be applied to DOT. We conjecture that, although the DOT model does not suffer the same deterioration in accuracy as the DOP model due to the probability estimation bias, better estimation of the fragment probabilities will, nevertheless, lead to further improvements in accuracy as the fragment set increases in size. Furthermore, we hypothesise that searching for the most probable translation may yield higher translation accuracy than searching for the shortest derivation if parameter estimation is improved.

6.5.6 Which translation direction is more difficult for DOT?

Looking at the average times per sentence and the average numbers of samples taken per sentence as shown in Tables 6.11 and 6.12, it appears that it is more difficult to translate from French to English than from English to French. This seems reasonable as the DOP experiments presented in section 3.2.5 showed that parsing the French section of the HomeCentre corpus was more difficult than parsing the English section. Looking, however, at translation accuracy, we see that translating from French to English also produces higher quality translations. For example, over all translations for all ranking strategies and at all fragment depths, the highest exact match score for French to English translation was 50.42% whereas the highest exact match score for English to French translation was 42.49%. This outcome illustrates a weakness of the DOT model, namely that it finds phenomena such as determiner-noun agreement, determiner-adjective-noun agreement and subject-verb agreement difficult to model – examples of translations produced which contain agreement errors are given in Table 6.20. Thus, it is not surprising that translating into French, for which these types of agreements are very common, proves more difficult than translating into English, for which they are less prevalent.

Source	in Windows 95 , click the Xerox Document HomeCentre button on **the taskbar** .
Reference	sous Windows 95 , cliquez sur le bouton Xerox Document HomeCentre figurant sur **la barre des tâches** .
DOT	sous Windows 95 , cliquez sur le Xerox Document HomeCentre bouton figurant sur **le barre des tâches** .
Source	in **the** Save As dialog **box** , type a name in **the** preset name **box** and click OK .
Reference	dans **la boîte** de dialogue Enregistrer sous , tapez un nom dans la zone Nom préréglage et cliquez sur OK .
DOT	dans **le boîte** de dialogue Enregistrer sous , type un nom dans **le boîte** qui contient la nom préréglage et cliquez sur OK .

Table 6.20: Examples of translations produced by the DOT model which contain agreement errors.

6.6 Acquisition of sub-structurally aligned bilingual treebanks

Our evaluation of the DOT model has shown that it achieves high levels of translation accuracy. However, the problem of data acquisition constitutes a serious bottleneck as DOT requires that parsed sentence pairs be aligned at sentential and sub-structural levels. Manually establishing sub-structural alignments – as was done for the data used in the experiments presented thus far – is impractical because it is time-consuming and requires considerable expertise of both source and target languages as well as how they are related; clearly, automation of this process is essential.

Groves et al. (2004) present an algorithm which automatically induces sub-structural alignments between context-free phrase structure trees in a fast and consistent fashion. This algorithm starts by finding lexical correspondences between the source and target trees and then proceeds from the aligned lexical terminal nodes in a bottom-up fashion, using a mixture of node label matching and structural information to link source and target node pairs within the trees.

In order to assess the performance of the DOT model when translational links are inserted automatically rather than manually, we present parallel DOT experiments (also presented in (Groves et al., 2004)). We took two identical copies of a set of 605 English-French sententially-aligned tree pairs from the HomeCentre corpus and aligned one of these copies manually (C_m) and the other (C_a) automatically. We then randomly extracted 8

(*A*)

	BLEU		F-Score	
	Auto	Man	Auto	Man
1	0.0605	**0.2627**	0.3558	**0.5506**
2	0.1902	**0.3018**	0.4867	**0.5870**
3	0.1983	**0.3235**	0.4957	**0.6045**
4	0.2140	**0.3235**	0.5042	**0.6069**

(*B*)

	BLEU		F-Score	
	Auto	Man	Auto	Man
1	0.6118	**0.6591**	0.7900	**0.8090**
2	**0.7519**	0.7144	**0.8751**	0.8446
3	**0.7790**	0.7610	**0.8887**	0.8688
4	**0.7940**	0.7611	**0.8930**	0.8736

Table 6.21: Results of English to French translation experiments using automatic and manual alignments; table (*A*) gives results over all translations and table (*B*) gives results for those sentences translated using both types of alignments.

different training/test splits from C_m and C_a such that C_m^i and C_a^i contained the same test sentences and training trees. As the only difference between the DOT experiments run over splits C_m^1–C_m^8 and C_a^1–C_a^8 is in the translational links between source and target training trees, this was the only possible reason for differences in translation accuracy. Thus, examination of the output translations allows us to establish the consequences of using automatic rather than manual alignments.

We present the results of these experiments in Table 6.21. Looking firstly at (*A*), we see that using the manually linked fragment base results in significantly better overall performance at all link depths than using the automatic alignments. However, the comparatively poor scores achieved using the automatically induced alignments reflect the fact that these alignments give poorer coverage at all depths than those determined manually (47.71% vs. 66.46% at link depth 1, 56.39% vs. 67.92% at link depths 2 - 4). The results in (*B*) – where we evaluated only the subset of sentences for which translations were produced both when the manually aligned fragment bases were used and when the automatically linked ones were used – show that the translations generated using automatic alignment are actually of comparable quality to those generated using manual alignments.

While the alignment algorithm used in these experiments is quite basic, our evaluation indicates clearly that using such an algorithm is a viable alternative to manual alignment. Furthermore, we believe that the development of a more sophisticated algorithm will yield even greater returns in terms of translation coverage and quality. Consequently, we feel that automatic acquisition of the resources required by the DOT model is a real possibility and deserves further attention.

6.7 Summary

Using the DOT system described in chapter 5, we have carried out a more detailed study of the performance of the DOT model of translation than presented heretofore. In this chapter, we have described our experiments in terms of the data used and the evaluation metrics upon which our assessment of the performance of the DOT model is based. Our evaluation shows firstly that, for DOT, translation accuracy is very high according to all evaluation metrics used. Even the worst DOT score (i.e. the combination of fragment depth and ranking method scoring lowest) for each metric is better than the score achieved using an SMT system trained and tested on the same data. Furthermore, the best DOT score for each metric scores significantly higher than the SMT system. In addition, translation times are faster than might have been assumed. Interestingly, the 'simplicity' model proposed for DOP does extremely well for DOT, outperforming the standard MPT model. We hypothesise that further improvements may well be gained by combining the two models, as was the case for DOP (Bod, 2003a). Finally, preliminary work on automatic resource acquisition for DOT has proved promising. This will be crucial if DOT systems are to be scaled up further.

The DOT model evaluated here operates solely on the level of syntactic structure. Consequently, many of the errors in translation output center around linguistic phenomena such as determiner-noun agreement, determiner-adjective-noun agreement and subject-verb agreement. However, the data-oriented approach to translation in general is not limited in terms of linguistic description to context-free phrase-structure trees. DOT models can also be defined for representations corresponding to more sophisticated linguistic formalisms. We discuss one such model, which assumes the representations of Lexical-Functional Grammar, in chapter 8.

Chapter 7

A richer DOP model: LFG-DOP

All DOP models assume a corpus of utterances which have been annotated with linguistic representations conforming to a particular grammar formalism, from which a set of fragments can be extracted and the probabilities of those fragments estimated. Consequently, the DOP output for any input string is an analysis which also conforms to the same grammar formalism used to annotate the corpus – if the corpus annotations are context-free phrase structure trees, as is the case for the Tree-DOP model described in chapter 2, then the analysis assigned to any input string will also be a context-free phrase structure tree. While experimental results show that the Tree-DOP model achieves excellent parse accuracy (e.g. (Bod, 2001, 2003a)), the expressive power of this model is nevertheless limited by the corpus representations it assumes. It is known that these representations, which reflect surface syntactic phenomena only, do not adequately describe many aspects of human language.

The LFG formalism (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001), on the other hand, is known to be beyond context-free. As its representations encode grammatical features (such as number, case and tense) and identify the grammatical functions of constituents (such as subject, object and complement) in the context in which they occur, they are powerful enough to express linguistic phenomena occurring at levels other than surface structure. Thus, there is clear motivation for the development of parsing models which output this type of linguistic analysis. A DOP model which assumes LFG representations was proposed by Bod and Kaplan (1998) and further refinements to the

model, as well as empirical evaluations, were published in (Bod, 2000a,b; Bod and Kaplan, 2003).

In section 7.1 of this chapter, we describe the proposed LFG-DOP model while in section 7.2 we outline the available details of the LFG-DOP implementation which has been built and give the results achieved using this system. In section 7.3, we discuss some of the implications of how LFG-DOP fragments have been defined and propose an alternative method of specifying fragment constraints. In section 7.4, we show how parameter re-estimation techniques developed for Tree-DOP using back-off can also be applied in the LFG-DOP model. Furthermore, we describe how the back-off relationships between fragments can be used to achieve robustness in an efficient manner. Finally, in section 7.5 we discuss the application of efficient Tree-DOP parsing and disambiguation strategies to the implementation of the LFG-DOP model.

7.1 The LFG-DOP Model

As for Tree-DOP, providing a specification of the LFG-DOP model means we must specify four elements: the type of representation we expect to find in the example base, how fragments are to be extracted from those representations, how extracted fragments are to be recombined when parsing new input strings and how the resulting parses are to be ranked. In the following sections, we provide details of each of these elements for the LFG-DOP model.

7.1.1 Representations

The representations found in the LFG-DOP example base are <c,ϕ,f> triples comprising c-structure, ϕ-links and f-structure which correspond to LFG theory – an example LFG representation is given in Figure 7.1. The c-structures take the form of context-free phrase-structure trees, while the f-structures are attribute-value matrices. The ϕ-links are mappings between c-structure nodes and f-structure values. F-structure values can be either simple or complex, where a complex value is an f-structure unit. For example, the outer f-structure unit f_1 in Figure 7.1 comprises six attributes; the value associated with the SUBJ attribute is a complex value, whereas the values associated with all the other

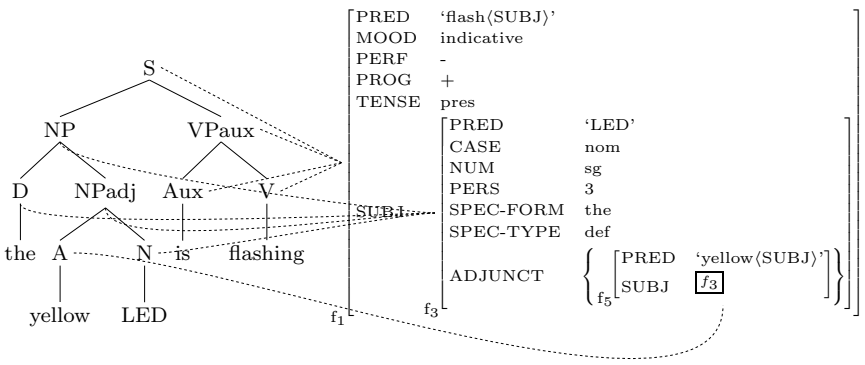
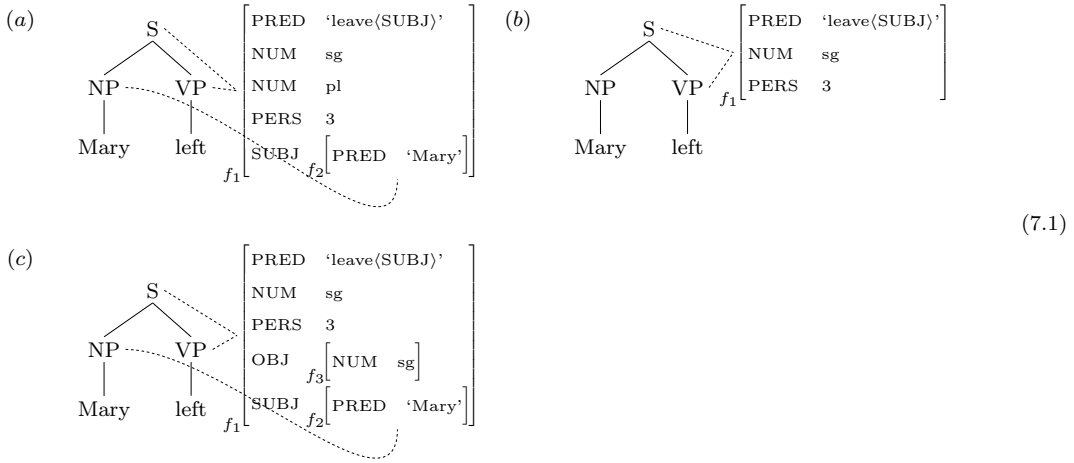


Figure 7.1: An example LFG representation for the string *the yellow LED is flashing.*

attributes are simple values.

For an f-structure to be well-formed according to LFG theory, each attribute must have exactly one value – this stipulation is called the *uniqueness* condition. Simple values comprising lemmas use argument lists to encode subcategorisation information about the c-structure terminals to which they correspond, i.e. they specify the grammatical functions required by the lexeme. For example, the value of the PRED attribute in f-structure unit f_1 of the example in Figure 7.1 stipulates that *flash* subcategorises for a subject only. According to the *completeness* condition, in order for the f-structure to be well-formed, a SUBJ attribute must also be present in the f-structure unit local to this surface form, i.e. unit f_1. Furthermore, the *coherence* condition requires that all governable attributes local to an f-structure unit must be subcategorised for by the surface form local to that f-structure unit. The representations in example (7.1) below are examples of ill-formed LFG representations: representation (*a*) violates the uniqueness condition as it contains two values for the feature NUM, representation (*b*) violates the completeness condition as the SUBJ required by *leave* is missing and representation (*c*) violates the coherence condition as the governable attribute OBJ is not subcategorised for by *leave*. The uniqueness and coherence conditions are monotonic, meaning that they will be unsatisfied for a superstructure if they are unsatisfied for any of its substructures. Completeness, on the other hand, is non-monotonic, i.e. completeness may be unsatisfied for a substructure but satisfied for any of its superstructures.



(7.1)

7.1.2 Fragmentation

The fragmentation operators for LFG-DOP are extensions of those used in Tree-DOP as we wish to extract exactly the same set of generalised c-structure fragments as before. However, we also wish to extract the corresponding f-structure fragment to go with each c-structure. Consequently, the original *root* and *frontier* operators must be extended to take f-structure into account. Many different extensions can be envisaged; those defined in (Bod and Kaplan, 1998, 2003; Bod, 2000a,b), are as follows:

Root Given a copy of the example-base representation <c,ϕ,f> named <c_{copy},ϕ_{copy},f_{copy}>:

1. select a node in c_{copy} to be *root* and delete all nodes except this node and the nodes it dominates;
2. delete all links in ϕ_{copy} which link deleted c-structure nodes to f_{copy};
3. delete all f-structure units in f_{copy} which are not ϕ-*accessible* from c_{copy};
4. delete all semantic forms in f_{copy} which are local to f-structure units corresponding to erased c-structure terminals.

Frontier Given a representation of the form <c_{copy},ϕ_{copy},f_{copy}> created by the root operation:

1. select a (possibly empty) set of nodes in c_{copy} to be *frontier* nodes and delete all nodes dominated by these newly-created frontier nodes;

2. delete all links in ϕ_{copy} which link deleted c-structure nodes to f_{copy};

3. *(delete all f-structure units in f_{copy} which are not* ϕ-accessible *from c_{copy};)*

4. delete all semantic forms in f_{copy} which are local to f-structure units corresponding to erased c-structure terminals.

Step 3 of the frontier operation (shown in brackets above) is given here for the sake of completeness; as a consequence of the definition of ϕ-accessibility given in (Bod and Kaplan, 1998, 2003; Bod, 2000a,b) and described below, the root node of c_{copy} (selected during the root operation) accesses all f-structure units in f_{copy} regardless of which nodes are selected by the frontier operation. This definition of ϕ-accessibility is as follows:

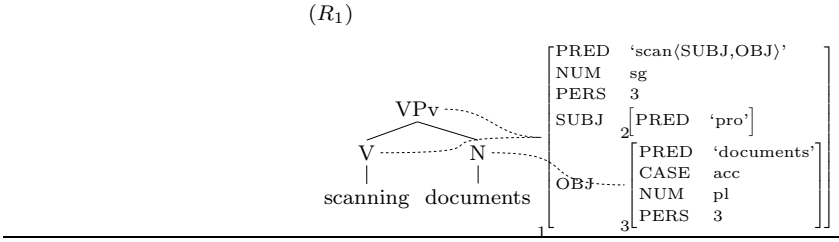
ϕ-accessibility An f-structure unit f is ϕ-accessible from c-structure node n if and only if

1. n is ϕ-linked to f, i.e. $\phi(n) = f$, or

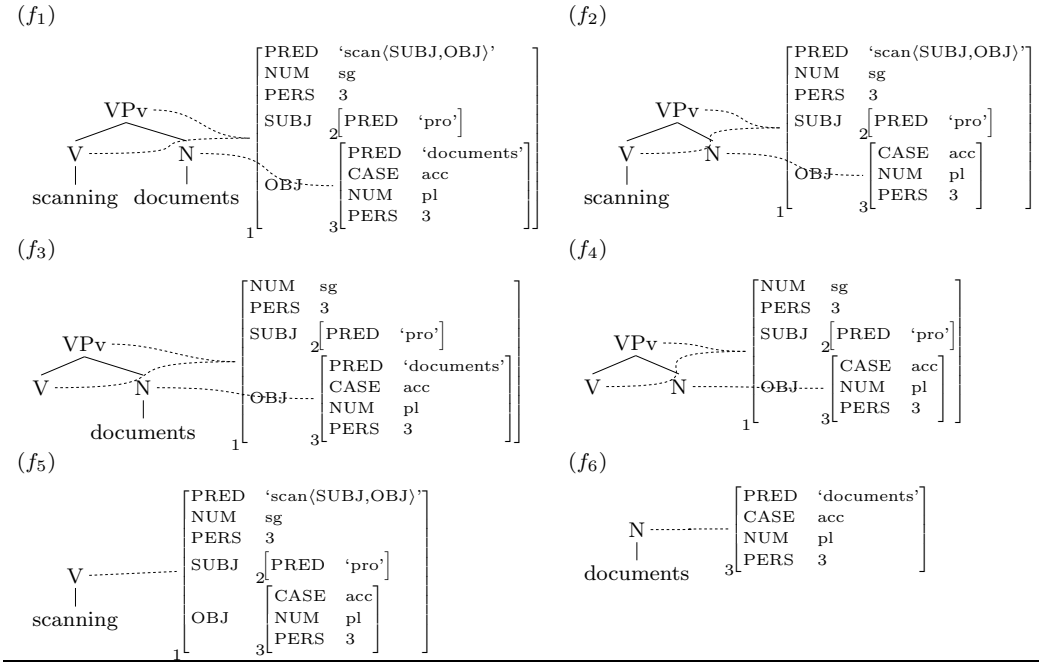
2. n is ϕ-linked to f_x and f contains f_x i.e. there is a chain of attributes leading from f to f_x.

The fragmentation process is illustrated in Figure 7.2 where, through the application of root and frontier, representation R_1 yields fragments f_1 – f_6. Consider, for example, the extraction of fragment f_6: during step 1 of the root operation, node N was selected to be root and all nodes not dominated by this node were deleted; during step 2 all ϕ-links specifying the deleted nodes VPv and V were deleted; during step 3 the f-structure units 1 and 2 were deleted as they were no longer ϕ-accessible from the c-structure and during step 4 no changes were made as there were no remaining lemmas corresponding to deleted c-structure terminal symbols. As step 1 of the frontier operation selected the empty set, no further representation elements were deleted during the application of frontier operation steps 2 – 4. In contrast, deletion of f-structure elements occurs solely during application of the frontier operation when extracting fragment f_4: during step 1 of the root operation, node VPv was selected as root, meaning that application of root steps 2 – 4 effected no changes to the f-structure. During step 1 of the frontier operation, nodes V and N were

(**A**) **A sample LFG representation** :



(**B**) **The LFG-DOP fragments which can be extracted from representation R_1 using root and frontier** :



(**C**) **The LFG-DOP fragments which can be extracted from fragment f_6 using discard** :

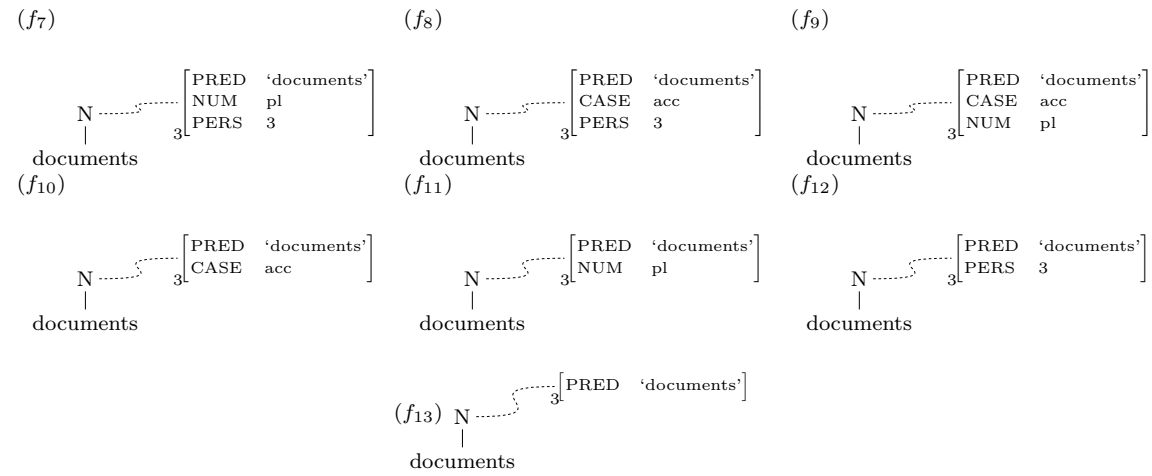


Figure 7.2: The LFG-DOP fragmentation process: application of the root and frontier operations to treebank representation T_1 yields the 6 fragments f_1–f_6; application of the discard operation to fragment f_6 yields fragments f_7–f_{13}.

selected to be frontiers and the subtrees they dominate deleted; during step 2, no ϕ-links are deleted as the only c-structure nodes deleted were terminal symbols; however, during step 4, the semantic forms corresponding to those deleted terminal symbols were erased from the f-structure.

The extended root and frontier operations for LFG-DOP yield precisely the same c-structures as are yielded by the root and frontier operations defined for Tree-DOP. However, it is also possible to extract further fragments from each fragment yielded by the root and frontier operations through use of the *discard* operation. Discard is used to delete attribute-value pairs from the f-structure whose values are not ϕ-linked to remaining nodes in the c-structure and are not surface-forms corresponding to c-structure terminals. Thus, when discard is applied to any fragment $<c,\phi,f>$, a new fragment $<c,\phi,f_{d_x}>$ is extracted; the c-structure and ϕ-links are unchanged but f_{d_x} differs from f in that all attribute-value pairs in f_{d_x} are also in f but the reverse does not hold, i.e. it is not the case that all attribute-value pairs in f are also in f_{d_x}.

Fragments f_7 to f_{13} in Figure 7.2(C) were extracted from fragment f_6 in Figure 7.2(B) using the discard operation. Fragment f_7, for example, was extracted by deleting the CASE attribute and its value, meaning that this noun fragment can now appear in any sentence position rather than being restricted to those positions taking accusative case. Fragment f_{13} was extracted by deleting all eligible attribute-value pairs – the attribute PRED is not eligible for deletion as its value corresponds to the c-structure terminal *documents* – and is, thus, the least specific f-structure which can be extracted from this fragment. The number of fragments DF extracted using discard from the fragment $<c,\phi,f>$ is exactly the number of ways attribute-value pairs can be deleted from f without violating the ϕ-links. This can be calculated using the formula given in (7.2) below where m is the number of attribute-value pairs in the outermost f-structure unit of f whose values are simple and not lemmas (including values which are indices referencing complex values), f_x is the complex value of an attribute-value pair in the outermost f-structure unit of f which is not linked to a c-structure node and f_y is the complex value of an attribute-value pair in the outermost f-structure unit of f which is linked to a c-structure node. (We subtract 1 to adjust for the fact that the fragment base already contains a fragment where

no attribute-value pairs have been deleted from the f-structure and thus we do not allow discard to return a fragment to which no deletions have been made.)

$$DF(f) = 2^m \prod_{f_x}(DF(f_x)+1) \prod_{f_y} DF(f_y) - 1 \tag{7.2}$$

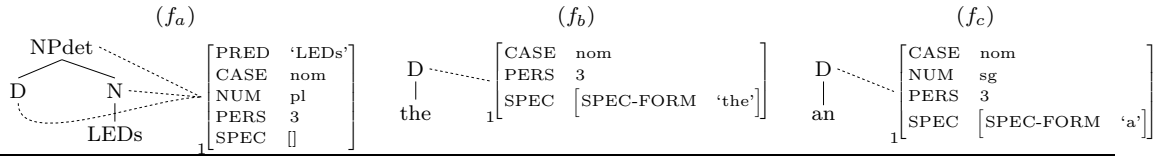
For example, according to this formula the discard operation extracts a further 95 fragments from fragment f_1 in Figure 7.2(B), each of which has the same c-structure and ϕ-links as fragment f_1 but differs with regard to the constraints imposed in its f-structure. Hence, the number of discard fragments which can be extracted from the set of fragments in Figure 7.2(B) is 482 and the total number of root, frontier and discard fragments which can be extracted from representation R_1 is 488.

7.1.3 Composition

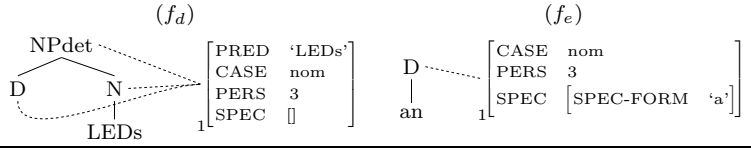
The LFG-DOP composition operation involves two stages: leftmost substitution over c-structure and recursive unification over f-structure such that the ϕ-links are not broken. This process is illustrated in Figure 7.3. C-structure composition is defined exactly as for Tree-DOP: derivation sequence der_1 in Figure 7.3(C), involving the composition of the c-structures of f_a and f_b, requires the substitution of f_b at the leftmost substitution site of f_a provided that the root node of f_b and the leftmost substitution site of f_a are of the same category. This is followed by recursive unification of f_b's f-structure with the f-structure unit of f_a to which the leftmost substitution site in f_a was linked according to the ϕ-links.

Any sequence of composition operations yielding a complete derivation (i.e. one which contains no open c-structure substitution sites) is only valid if that derivation's f-structure adheres to the LFG well-formedness conditions described in section 7.1.1. However, the presence in the fragment base of discard-generated fragments means that many apparently ill-formed input strings can also be parsed. This process is illustrated in derivations der_2 – der_4 in Figure 7.3(C) where fragments formed using root and frontier only (i.e. f_a, f_b and f_c) and fragments formed using root, frontier and discard (i.e. f_d and f_e) can be used unrestrictedly in deriving analyses of the ill-formed input *an LEDs*. The formation of an analysis by composing fragments f_a and f_c yields an f-structure which is invalid due to

(**A**) **A sample fragment base generated using root and frontier** :



(**B**) **Some discard fragments which can be generated from** (**A**) :



(**C**) **Some composition sequences using fragments from** (**A**) **and** (**B**) :

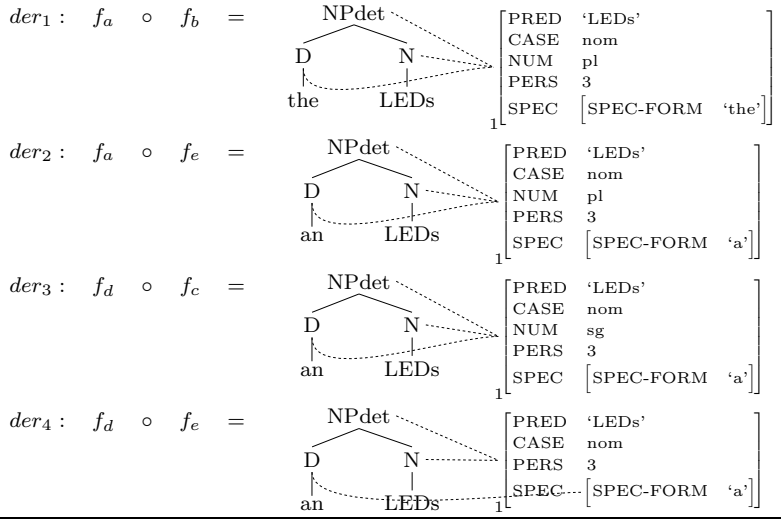


Figure 7.3: Fragments f_d and f_e were extracted using discard from the root and frontier fragments f_a and f_c. The ungrammatical input *an LEDs* can be assigned analyses using discard fragments from which the attribute-value pairs causing clashes have been deleted.

the presence of two values for the attribute NUM as f_a is plural whereas f_c is singular. In der_2, however, fragment f_e – extracted by deleting the NUM attribute from fragment f_c – is composed with f_a to yield a plural analysis for the input, and in the second, fragment f_d – extracted by deleting the NUM attribute from fragment f_a – is composed with f_c to yield a singular analysis. The third derivation, in which fragments f_d and f_e – both of which are generated by deleting NUM attributes – are composed, yields an analysis for the input in which NUM is not specified.

Thus, the use of discard fragments gives a robust system as analyses can be provided for many input strings perceived to be ungrammatical. The notion of grammaticality is retained, however, through the following definition (Bod and Kaplan, 2003):

> A sentence is *grammatical with respect to the corpus* if and only if it has at least one valid representation with at least one derivation without generalised fragments.

Thus, strings which can only be parsed using one or more discard-generated fragments are deemed ungrammatical with respect to the corpus; this accounts for both ill-formed strings and strings which are well-formed but beyond the scope of the training data.

7.1.4 The Probability Model

In the Tree-DOP model, valid derivations are constructed by composing fragments such that the category-matching condition is fulfilled. Each valid derivation is assigned a probability by calculating the product of the probabilities of the fragments used in the construction of that derivation. The probabilities of all valid derivations which can be constructed from a given DOP grammar for all of the strings which it recognises sums to 1.

Each LFG-DOP derivation probability is also calculated as the product of the probabilities of the fragments used in the construction of that derivation. In the LFG-DOP model, however, we have seen that valid derivations are constructed by composing fragments such that the category-matching, uniqueness, completeness and coherence conditions are fulfilled. If we calculate LFG-DOP fragment probability distributions in the same way as we did for Tree-DOP – i.e. define distributions over root node category – then the probabilities of all derivations for all the strings recognised by the grammar which adhere

to the category-matching condition will sum to 1. However, it is not the case that all of these derivations are valid according to the LFG-DOP model as they may not fulfil the uniqueness, completeness and coherence conditions. Consequently, the probabilities of all *valid* derivations which can be constructed from a given LFG-DOP grammar for all of the strings which it recognises no longer sums to 1 and, therefore, does not constitute a probability distribution. In other words, the LFG-DOP model 'leaks' probability mass by assigning non-zero probabilities to derivations which are not actually valid.

In order to present the LFG-DOP probability model proposed by (Bod and Kaplan, 1998, 2003) (under which the probabilities of all valid derivations of all recognised strings sums to 1), it is helpful to restate the Tree-DOP probability model as follows. Building a Tree-DOP derivation can be viewed as a top-down stochastic branching process. A fragment whose root node corresponds to the start category is selected at random to start the derivation. Further fragments are successively chosen to combine with the leftmost open substitution site of the derivation; these fragments are chosen at random from the set of fragments competing for selection at each substitution site. Thus, the competition probability (CP) of selecting a fragment at random to participate in a derivation is the likelihood with which it is drawn from the competition set (CS), i.e. its probability over the total probability mass assigned to the CS as given in equation (7.3).

$$CP(f) = \frac{P(f)}{\sum_{f' \in CS} P(f')} \tag{7.3}$$

The only criterion which fragments must meet in order to belong to a given competition set is that their root nodes are of the same syntactic category as the leftmost substitution site in question. Consequently, the distribution of the competition sets corresponds exactly to the distribution of fragments in the fragment base, and dividing by the total probability mass assigned to each competition set is unnecessary in practice.

As many Tree-DOP derivations can yield the same parse tree, the probability of a parse is the sum of the probabilities of the derivations which yield that parse conditioned on the total probability mass assigned to those randomly-selected derivations which yield

valid parses as given in equation (7.4).

$$P(T|T\ is\ valid) = \frac{P(T)}{\sum_{T'\ is\ valid} P(T')} \tag{7.4}$$

Again, as the only condition imposed on fragment combinations is that of category matching, all random derivations yield valid parse trees and it is unnecessary in practice to normalise by dividing by the probability mass assigned to valid parses.

Bod and Kaplan (1998, 2003) make the observation that the process of building an LFG-DOP derivation can also be viewed as a top-down stochastic branching process but this process differs from that of Tree-DOP in two crucial ways. Firstly, not all fragments whose root nodes correspond to the leftmost substitution site category of a partial derivation are members of the competition set and, therefore, eligible for composition with the derivation at hand as, as previously stated, the category-matching condition is not the only well-formedness condition for LFG-DOP. This means that the probability distribution of each competition set is no longer the same as the distribution of fragments in the fragment base and we must condition fragment selection probability on the competition set probability mass as in equation (7.3). Recall, however, that the completeness condition is non-monotonic. This means that no partial derivation can be checked for completeness, i.e. a derivation can only be judged for completeness when it contains no open substitution sites and, therefore, no further fragments can be composed with it. This characteristic of LFG derivations results in the second crucial difference between the LFG-DOP and Tree-DOP models, namely that the stochastic branching process by which derivations are constructed does not necessarily yield valid representations even when the other well-formedness conditions have been verified during fragment selection. Consequently, it is necessary to normalise in terms of the probability mass assigned to valid derivations only as specified in equation (7.4).

While the completeness condition must be enforced after sampling, the uniqueness and coherence conditions can be enforced either during or after sampling. Accordingly, three of the many possible probability models – M_1, M_2 and M_3 – corresponding to three different competition set definitions are described by Bod and Kaplan (1998, 2003).

Model $\mathbf{M}_1$ Model M_1 is the simplest model as it is a straightforward extension of the Tree-DOP model. That is, only the category matching condition is enforced during sampling; after sampling, each derivation is checked for uniqueness, completeness and coherence and deemed either valid or invalid accordingly. Therefore, having completed derivation step D_{i-1}, the next competition set CS_i is the set of fragments whose root nodes match the leftmost substitution site LSS of D_{i-1}. The competition sets for Model M_1 are thus calculated according to the definition given in (7.5).

$$CS_{M_1} = \{f : root(f) = LSS(D_{i-1})\} \tag{7.5}$$

Model $\mathbf{M}_2$ Model M_2 defines the competition sets so that the uniqueness condition is checked at each sampling step. Only fragments with the appropriate root node and which unify successfully (i.e. without introducing clashes) with the current partial derivation are included in the competition set; after sampling, each completed derivation is checked for completeness and coherence and deemed either valid or invalid accordingly. The competition set for Model M_2 having completed derivation step D_{i-1} is thus calculated according to the definition given in (7.6).

$$CS_{M_2} = \{f : root(f) = LSS(D_{i-1}) \wedge unique(D_{i-1} \circ f)\} \tag{7.6}$$

Model $\mathbf{M}_3$ Model M_3 defines the competition sets such that both the uniqueness and coherence conditions are checked each time a fragment is composed with the current subderivation. Consequently, only fragments with the appropriate root node and which unify without violating either the uniqueness or coherence conditions are included in the competition set. Again, the completeness condition is verified only after the full derivation has been sampled. The competition set for Model M_3 having completed derivation step D_{i-1} is calculated according to the definition given in (7.7).

$$CS_{M_3} = \{f : root(f) = LSS(D_{i-1}) \wedge unique(D_{i-1} \circ f) \wedge coherent(D_{i-1} \circ f)\} \tag{7.7}$$

Estimating fragment probabilities Each of the models M_1 – M_3 described above requires an initial estimate for the probability of each LFG-DOP fragment. One possible way to estimate the probability of a Tree-DOP fragment is to take its empirical frequency conditioned on root node. This estimator can also be applied to LFG-DOP fragments – it is termed 'simple relative frequency' or 'simple RF' by Bod and Kaplan (2003). However, this estimator draws no distinction between fragments generated by the root and frontier operators and discard-generated fragments. If the number of fragments which can be extracted from a treebank using the root and frontier operations is generally very large, the number of discard-generated fragments which can be extracted from the set of fragments generated by root and frontier is far larger again. Recall that application of the root and frontier operators to the representation R_1 in Figure 7.2 yielded 6 fragments in contrast to the 482 fragments yielded by discard. In other words, almost 99% of the fragment space is occupied by those fragments generated by the discard operation. As the simple RF estimator treats all fragments equally regardless of how they were generated, this means that almost 99% of the probability mass is also given to these fragments. Fragments generated by discard effectively relax the constraints specified in the f-structure to allow the fragment to be used in a wider variety of contexts and so are very useful in constraint-based DOP parsing. Intuitively, however, they should only be considered when no parse can be produced which satisfies all relevant constraints, i.e. they should be used only when the input is ill-formed. Consequently, this estimator does not seem entirely appropriate.

Bod and Kaplan (2003) present an alternative method – termed 'discounted RF' – of estimating fragment probabilities whereby root and frontier fragments are treated as seen events and discard fragments as unseen events. The fragment set is partitioned using this distinction and two separate probability distributions induced. The probabilities of seen events are estimated by their relative frequencies as before. However, these probabilities are then discounted and the discounted mass distributed amongst the unseen events. The amount of probability mass to be discounted from seen events is calculated using the Good-Turing estimator which computes the probability mass to be discounted as $\frac{n_1}{N}$ where n_1 is the number of fragments occurring just once in the set of seen events and N is the total number of seen events. Thus, the total probability mass assigned to seen fragments is

reduced from 1 to $1 - \frac{n_1}{N}$ as shown in equation (7.8) and the unseen fragments are assigned probabilities proportional to the remaining mass $\frac{n_1}{N}$ as shown in equation (7.9).

$$P(f : f \in seen) = (1 - \frac{n_1}{N}) \frac{|f|}{\sum_{f' \in seen} |f'|} \tag{7.8}$$

$$P(f : f \in unseen) = (\frac{n_1}{N}) \frac{|f|}{\sum_{f' \in unseen} |f'|} \tag{7.9}$$

As the discount RF estimator assigns a fixed amount of probability mass to the discard-generated fragments, the fact that the number of discard fragments is far greater than the number of root and frontier fragments no longer affects their probabilities.

Discussion

In order to ensure that the probabilities of all valid LFG-DOP derivations which can be constructed for a given LFG-DOP grammar for all of the strings which it recognises sums to 1, models M_1 – M_3 specify that fragment probabilities are normalised. Each fragment probability is conditioned on the probability mass of the set of valid fragments from which it is selected to participate in the derivation at hand and the resultant probability of each derivation produced is conditioned on the probability mass of the set of valid derivations. This normalisation procedure would appear to rectify the problem of probability mass being assigned to invalid representations by ensuring a proper probability distribution over valid derivations. However, Abney (1997) observes that normalisation serves only to mask the fact that, unlike for the context-free case, establishing probabilities for grammars encoding context-sensitive dependencies using relative frequency estimation does not yield the best weights. Bod and Kaplan (2003) note that neither of the methods they propose for estimating fragment probabilities – simple RF and discounted RF – address this issue, and that determining what kind of estimator is the true one will form part of future research.

7.2 LFG-DOP in practice

Empirical evaluations of the LFG-DOP model have been carried out with two LFG-annotated corpora, both of which were annotated at Xerox PARC. These are the Verbmobil corpus, which comprises 540 parses, and the English HomeCentre corpus which comprises 980 parses. Each corpus was split randomly into 90% training sets and 10% test sets in 10 different ways such that all words in the test set were also contained in the training set. The sentences in each test set were then parsed using the representations in the corresponding training set according to the LFG-DOP model. Due to memory constraints the fragments extracted from each training set were limited to those of depth 4 or less.

In this section, we present these evaluations of the LFG-DOP model (Bod, 2000a,b; Bod and Kaplan, 2003) in terms of parsing architecture (section 7.2.1), evaluation methodology (section 7.2.2) and results achieved (section 7.2.3).

7.2.1 Parsing with LFG-DOP

As LFG representations comprise context-free phrase-structure trees (c-structure) associated with sets of constraints (f-structure) on the contexts in which these trees may occur, it is the c-structures which drive the parsing process for any input string. Consequently, the Tree-DOP parsing algorithms described in section 2.4 can be used to build the c-structure parse space for LFG-DOP. Cormons (1999) describes a method to assign an index to each node in each c-structure such that the index refers to the corresponding ϕ-linked f-structure unit; using this method, retrieval of the f-structure associated with each c-structure in the parse space is easily accomplished.

The LFG-DOP parser evaluated in (Bod, 2000a,b; Bod and Kaplan, 2003) uses the Tree-DOP method described in section 2.4.1 (whereby each fragment is converted to a rewrite rule of the form $root(f) \longrightarrow frontier_1(f)\ ...\ frontier_n(f)$) to compute the c-structure parse space for each input string, and then uses Cormons' indexing method to retrieve the corresponding f-structures.

Random sampling of derivations, as described in section 2.5.1, is used to approximate the LFG-DOP MPP, although the exact method of computing sampling probabilities is not given. However, the architecture employed corresponds to model M_3 as described in section

7.1, i.e. category-matching is enforced during parse-space computation and uniqueness and coherence during disambiguation. Consequently, fragments chosen to compose with any sub-derivation are selected from the set of fragments which can be composed with that sub-derivation without violating these conditions. (Bod, 2000a,b; Bod and Kaplan, 2003) state that as derivations are sampled in top-down, left-to-right order, *"the competition sets of composable fragments are computed on the fly ... by grouping the f-structure units that unify and that are coherent with the subderivation built so far"*. As each derivation can only be deemed incomplete once the entire derivation has been seen, incomplete derivations and their probabilities are simply not included in the sampling distribution. The number of samples to be taken is calculated according to the method described in section 2.5.1 and the maximum number of samples is set to 10,000.

7.2.2 Evaluating LFG-DOP output

Parser output is generally evaluated by comparing the analysis assigned by the parser to each test string to the reference parse provided for that test string in the annotated corpus. Parser output taking the form of context-free phrase-structure trees can be scored using the exact match metric – parses score 1 if they are identical to the reference parse and 0 otherwise – as well as by precision, recall and f-score metrics. These metrics compare the constituents present in the output parse with those present in the reference parse. A constituent is a syntactic category label occurring in a parse tree which spans a consecutive sequence of words in the input string and a constituent is correct if there is a corresponding node in the reference parse with the same syntactic category label spanning the same consecutive sequence of input string words. Precision, recall and f-score are calculated according to equations (3.5), (3.6) and (3.7) on page 71.

As observed by (Bod, 2000a,b; Bod and Kaplan, 2003), it is not immediately obvious how to best extend this metric to evaluate LFG representations. They chose a simple extension of the notion of 'correct constituent' which results in quite a harsh evaluation metric: a constituent in P is correct if there exists a constituent in T which has the same node label, spans the same sequence of input tokens and ϕ-corresponds to the same f-structure unit.

7.2.3 Current LFG-DOP performance

Empirical evidence supports the decision to estimate the probability distributions of root and frontier fragments and the distributions of discard fragments separately using the 'discounted RF' approach (as opposed to the 'simple RF' method where no distinction is made between fragment types). Exact match accuracy on the Verbmobil corpus jumped from 1.1% to 35.9% and from 2.7% to 38.4% on the HomeCentre corpus when the discounting method was used; precision and recall also increased dramatically. This indicates that treating generalised fragments (i.e. discard-generated fragments) in the same way probabilistically as ungeneralised fragments (i.e. root and frontier fragments) is harmful. Bod and Kaplan (2003) also observe that the inclusion of discard fragments in the parse space results in only a very slight increase in parse accuracy; of course, they remain crucial when parsing sentences which are ungrammatical with respect to the corpus. All further experiments outlined here include discard fragments in the parse space and estimate fragment probabilities using the discounted RF method.

Experiments assessing the impact of limiting the size of the LFG-DOP fragment base (Bod, 2000a,b) show that the DOP hypothesis holds for LFG-DOP: as larger fragments are included in the parse space, parse accuracy increases. Accuracy increased significantly as input was parsed using fragments of depth 1, 2 or less, 3 or less and 4 or less for both the Verbmobil and HomeCentre corpora. Exact match, precision and recall increased from 30.6%, 74.2% and 72.2% respectively at depth 1 to 35.9%, 75.5% and 76.4% at depth 4 for the Verbmobil corpus and from 31.3%, 75.0% and 71.5% respectively at depth 1 to 38.4%, 80.0% and 78.6% at depth 4 for the HomeCentre corpus. This is an important result because, given the relative linguistic sophistication of the representations, it could have been the case that maximal parse accuracy was achieved using only depth 1 fragments.

The usefulness of the presence of functional information when predicting correct tree structures was measured by evaluating the c-structures output by the LFG-DOP parser and comparing the results to the output of a Tree-DOP parser on the same data (Bod, 2000a,b; Bod and Kaplan, 2003). Evaluation was based on exact match and the precision, recall and f-score metrics standardly used when evaluating tree structures, described in section 3.2.2. The results indicate that the accuracy of the output tree structures improves

when functional information is also available. Tree structure accuracy on the Verbmobil corpus increased from 46.6% to 50.8% for exact match, 88.9% to 90.3% for precision and 86.7% for to 88.4% recall. Similarly, tree structure accuracy on the HomeCentre corpus increased from 49.0% to 53.2% for exact match, 93.4% to 95.8% for precision and 92.1% to 94.7% for recall.

7.3 On the nature of LFG-DOP fragments

The definitions of the root and frontier operations for LFG-DOP are identical to those for Tree-DOP in terms of c-structure fragment extraction; the LFG-DOP operators must also, however, associate an f-structure fragment with each c-structure fragment extracted. Bod and Kaplan (2003) specify f-structure fragments in terms of ϕ-accessibility as described in section 7.1.2 above. Using the notion of ϕ-accessibility to determine the set of constraints associated with each c-structure fragment is problematic for two reasons. Firstly, it does not adequately describe how circular and re-entrant structures – which occur frequently in real data – are to be handled and, secondly, it results in the retention of attribute-value pairs not warranted by the corresponding c-structure. In this section, we discuss each of these issues in turn and provide an alternative specification for LFG-DOP fragment extraction which addresses these issues.

Circular structures occur frequently in LFG representations. Consider, for example, the representation in Figure 7.4, where the noun *LED* is modified by the adjective *yellow*. In the f-structure, we see that *yellow* functions as an adjunct to *LED* while *LED* functions as the subject of *yellow*. This is indicated by co-indexation: the value of the SUBJ attribute in f-structure unit f_3 is the index of the outer f-structure unit, f_2. By selecting c-structure node A to be root using the root operation and the empty set of frontier nodes, we arrive at the depth 1 c-structure fragment representing $A \longrightarrow yellow$. However, the definition of ϕ-accessibility given does not indicate unambiguously exactly which f-structure units correspond to this fragment. Recall that f-structure unit f_x is retained if it is ϕ-linked to a node in the c-structure. Thus, for the fragment with root node A, f-structure unit f_3 is retained. Furthermore, the definition also states that if f is ϕ-linked to a node in the c-structure and f contains f_x then f_x is also retained, i.e. if a chain of attributes

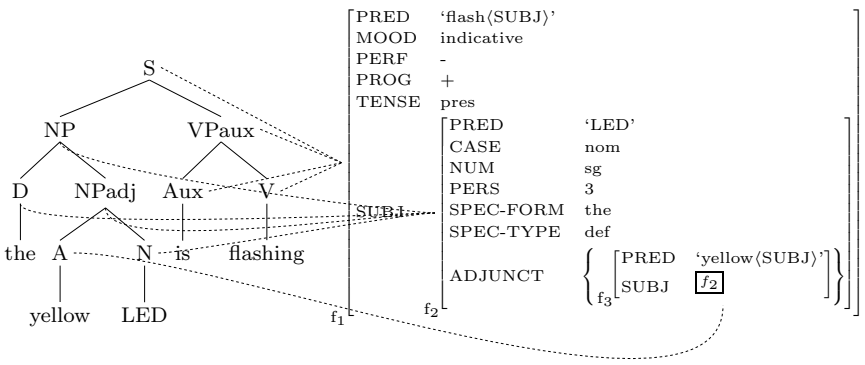


Figure 7.4: An example LFG representation for the string *the yellow LED is flashing.*

leads from f to f_x then f_x is retained. However, where circular f-structures are concerned, these two statements – intended to be unambiguous – are, in fact, contradictory. Clearly, unit f_3 in Figure 7.4 does not contain f-structure unit f_2. However, a chain of attributes *does* lead from f_3 to f_2: the value of f_3's SUBJ attribute is f_2. Thus, using the definition of ϕ-accessibility given by Bod and Kaplan (2003), we cannot establish the f-structure corresponding to fragment $A \longrightarrow yellow$.

Precisely the same issue arises for re-entrant structures. Consider, for example, the LFG representation given in Figure 7.5, where the SUBJ attribute and its value, subcategorised for by the infinitival verb *tomber*, are not contained within the same f-structure unit as the PRED attribute whose value is the lemma 'tomber'. Here, by selecting the c-structure node V dominating *tomber* to be root using the root operation and the empty set of frontier nodes, we arrive at the depth 1 c-structure fragment representing $V \longrightarrow tomber$. As before, the definition of ϕ-accessibility given does not indicate unambiguously exactly which f-structure units correspond to this fragment.

The definition of ϕ-accessibility also leads to the retention of f-structure units in LFG-DOP fragments for which it is questionable as to whether there is sufficient c-structure evidence to warrant their inclusion. Every f-structure unit in each representation is ϕ-accessible from the outermost f-structure unit because a chain of attributes leads from this outermost unit to every other unit in the f-structure. Consequently, all c-structure fragments containing at least one node which is ϕ-linked to the outermost f-structure

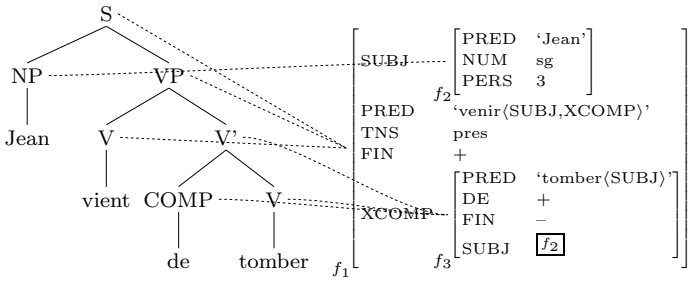


Figure 7.5: An example LFG representation for the French string *Jean vient de tomber*, which translates into English as *John just fell*, taken from (Way, 2001).

unit access *all* f-structure units. Thus, the fragment given in Figure 7.6(a), yielded by the representation in Figure 7.4, specifies that the subject of the verb *flashing* must have an adjunct. Furthermore, the constraints imposed on the depth 1 fragment with root node S are almost identical, only differing in that the PRED corresponding to the verb *flashing* is deleted. This fragment is shown in Figure 7.6(b). We believe that imposing such constraints is inappropriate given the evidence presented in the c-structure.

An alternative method of specifying which f-structure units are appropriate to each c-structure fragment, based on the notion of *support*, is discussed in (Bod and Kaplan, 1999) and (Way, 2001). According to this definition, all f-structure units *supported* by one or more c-structure nodes are retained. The definition of support is given in (7.10):

F-structure unit u is supported by node n if and only if:

(a) u is ϕ-linked to n, or

(b) u' is ϕ-linked to n **and**
u is the value of a grammatical-function attribute in u', or (7.10)

(c) u' is supported by n **and**
u is the value of a non-grammatical-function attribute in u' **and**
u is not ϕ-linked.

This definition of support facilitates the correct extraction of fragments comprising circular f-structure units. Consider the extraction of the fragment with root node A from the representation Figure 7.4. According to clause (b) of the support definition in (7.10), as

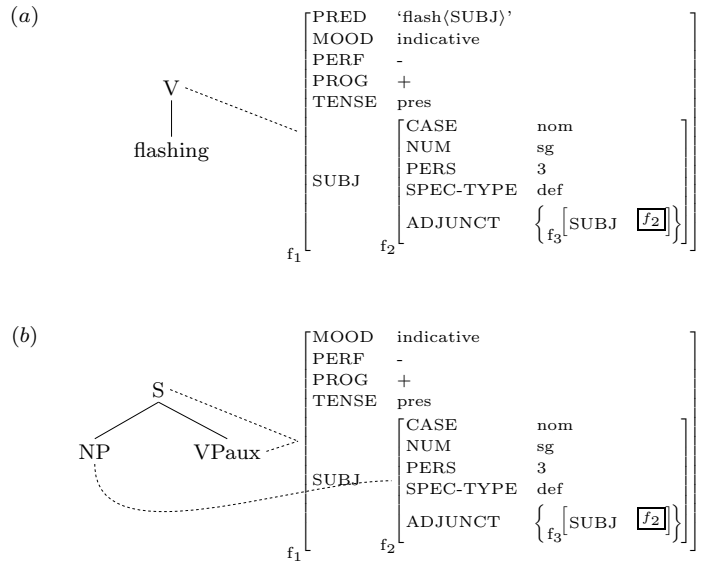
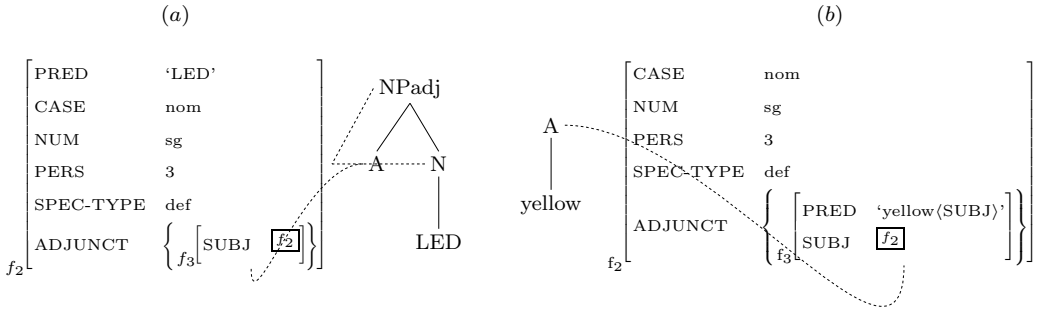


Figure 7.6: LFG-DOP fragments extracted from the LFG representation given in Figure 7.4 where the constraints present in the f-structure are determined using the accessibility criterion.

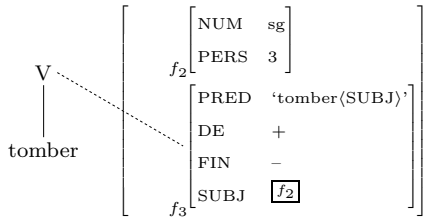
f-structure unit f_2 is the value of the grammatical-function attribute SUBJ in f_3 and f_3 is ϕ-linked to node A, unit f_2 is also supported. Thus, the LFG-DOP fragment corresponding to $A \longrightarrow$ *yellow*, given in (7.11)(b), specifies that the adjective *yellow* must occur as the modifier of a singular noun of nominative case such as the one given in (7.11)(a).



(7.11)

The definition of support does not, however, handle re-entrant structures in a satisfactory manner. By selecting the c-structure node V dominating *tomber* to be root using the root operation and the empty set of frontier nodes from the representation in Figure 7.5, we arrive at the depth 1 c-structure fragment representing $V \longrightarrow$ *tomber*. Using the

definition of support given in (7.10), we arrive at the LFG-DOP fragment given in example (7.12). These f-structure units do not specify the appropriate context in which *tomber* can occur as it is not clear which outer f-structure unit f_2 must unify with.



(7.12)

Furthermore, the definition of support, again, does not always remove f-structure information unwarranted by the c-structure fragment. Consider, for example, the fragments shown in Figure 7.7; they have identical c-structures to those shown in Figure 7.6 but the corresponding f-structures were determined using the notion of support rather than ϕ-accessibility. Looking firstly at Figure 7.7(a), we see that the c-structure fragment does not explicitly require an adjunct to the subject. The value of the SUBJ attribute (f_3) is supported according to clause 2 because this attribute is specified by the f-structure unit f_2 linked to the c-structure fragment. However, within unit f_3 the values of the non-grammatical-function attributes are supported (according to clause 3) but the grammatical-function attributes are not. Hence, the adjunct to the subject is not specified. Intuitively, this is as desired because the ADJUNCT attribute is an ungovernable grammatical function and is not subcategorised for by any element of the c-structure. Consider, however, the fragment in Figure 7.7(b). Here, the ADJUNCT attribute *is* supported because f-structure unit f_2 is ϕ-linked to c-structure node NP. However, we do not see that the requirement that the subject of any sentence be modified by an adjunct is warranted, given the evidence in the c-structure.

We suggest an alternative method of determining precisely the f-structure associated with each c-structure fragment. This method differentiates between governable and non-governable grammatical functions and proceeds as follows.

1. Determine a c-structure fragment using the root and frontier operations as for Tree-DOP (as before) but retain the full f-structure which appeared in the original representation.

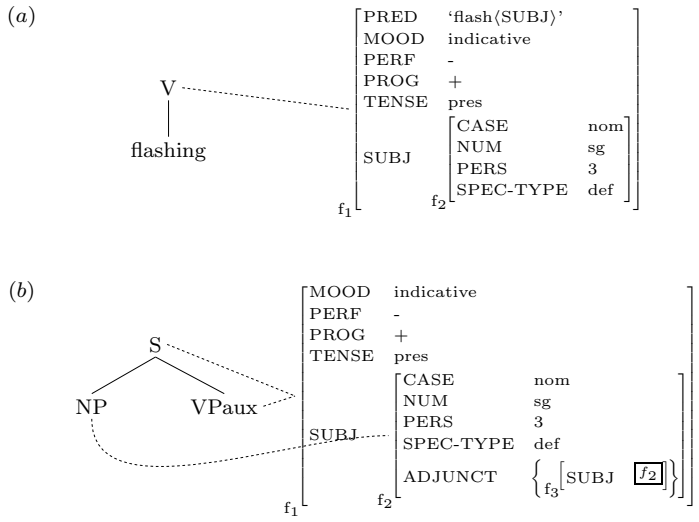


Figure 7.7: LFG-DOP fragments extracted from the LFG representation given in Figure 7.4 where the constraints present in the f-structure are determined using the support criterion.

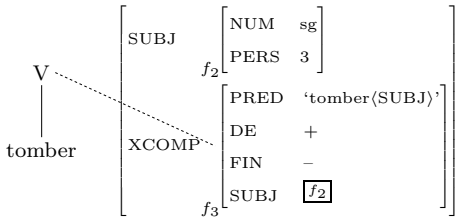
2. Delete all f-structure units (and the attributes with which they are associated) which are not ϕ-linked from one or more remaining c-structure nodes *unless* that unit is the value of an attribute subcategorised for by a PRED value whose corresponding terminal is dominated by the current fragment root node in the original representation.

   (a) Where we have floating f-structure units – i.e. a fragment is associated with f-structure units f_x and f_y such that f_x does not contain f_y and f_y does not contain f_x – then we also retain the minimal f-structure unit which contains them both. By minimal unit we mean the unit comprising the attribute with value f_x and the (nested sequence of minimal units containing) attribute with value f_y.

3. Delete all semantic forms (including PRED attributes and their values) not associated with one of the remaining c-structure terminals.

Example (7.13) shows the LFG-DOP fragment yielded by the representation given in Figure 7.5 for $V \longrightarrow$ *tomber* using this new definition of LFG-DOP fragment extraction.

Here, we see that the appropriate contextual information is specified: *tomber* must function as an XCOMP and its subject must correspond to the subject of the sentential unit which immediately dominates it.



(7.13)

The distinction drawn between governable and non-governable attributes is also important. F-structure units which are the values of non-governable attributes such as adjuncts are automatically deleted unless ϕ-linked to remaining c-structure nodes. Consider, for example, LFG-DOP fragment (a) in Figure 7.8 which was again extracted from the LFG representation given in Figure 7.4. As A is the only node linked to the value of the ADJUNCT attribute and this node is not present in the c-structure, both the ADJUNCT attribute and its value are deleted. Fragment (b) in Figure 7.8, however, illustrates the importance of distinguishing between governable and non-governable attributes: although no remaining c-structure node is linked to the value of the SUBJ attribute, this attribute-value pair is retained as it is subcategorised for by the PRED value 'flash<SUBJ>'. This example also illustrates the importance of not deleting lemmas until the appropriate f-structure units have been established. We do not know that all features of attributes which are subcategorised for are determined by the terminal corresponding to the PRED value – in this case, the auxiliary verb *is* carries crucial information regarding the subject of the sentence. This subject information is retained because the PRED value 'flash<SUBJ>' is not deleted until after the required f-structure units have been identified.

7.4 Parameter estimation for LFG-DOP

As discussed in section 2.6, computing fragment probabilities for Tree-DOP by estimating their relative frequencies in the fragment base introduces a bias towards larger parse trees. For LFG-DOP, two parameter estimation methods – described in section 7.1.4 – have been proposed. The first method estimates fragment relative frequencies without differentiating

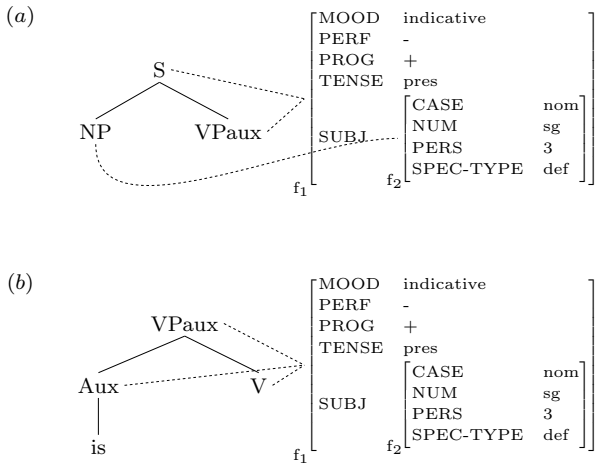


Figure 7.8: LFG-DOP fragments extracted from the LFG representation given in Figure 7.4 where the constraints present in the f-structure are determined by distinguishing between governable and non-governable attributes.

between fragments generated by root and frontier and fragments generated by discard. The second method distinguishes these two different types of fragments and estimates their relative frequencies separately. Furthermore, it involves placing a strict limit on the proportion of the probability mass assigned to discard-generated fragments as they are considered to be *unseen* events. Both of these estimation methods are, however, still based on the relative frequency estimator (where conditioning is on the root node of each c-structure) and, thus, still induce a bias in favour of larger parse trees just as for Tree-DOP. Here, we show how the structured approach to probability estimation developed for Tree-DOP can also be applied in the LFG-DOP model. Furthermore, we describe how the back-off relationships between fragments can be used to motivate the inclusion of discard-generated fragments in the parse space in an efficient manner.

7.4.1 Probability re-estimation using Back-off: LFG-DOP$_{bkf}$

In section 2.6.4, we described the definition of a parameter estimation method which takes into account the relationships between overlapping fragments given by Sima'an and Buratto (2003). Essentially, this approach involves organising the fragment base into a

hierarchically-structured space of correlated events according to the *back-off* relationship and re-estimating fragment probabilities according to this hierarchy. In (Hearne and Sima'an, 2003), we show that this methodology is also appropriate to the computation of LFG-DOP fragment probabilities and that, furthermore, it provides a natural solution to some of the difficulties of employing discard-generated fragments when parsing ill-formed input.

Back-off parameter estimation can be applied to LFG-DOP fragments generated by the root and frontier operations exactly as described for Tree-DOP, using a directed acyclic graph to represent the partial order between them. A directed edge points from a fragment $<c_x, \phi_x, f_x>$ to a pair of fragments $<<c_y, \phi_y, f_y>,<c_z, \phi_z, f_z>>$ if $<c_y, \phi_y, f_y>$ and $<c_z, \phi_z, f_z>$ compose to give $<c_x, \phi_x, f_x>$, as expressed in equation (7.14):

$$<c_x, \phi_x, f_x> \quad \geq_{bkf} \quad <c_y, \phi_y, f_y> \quad \circ \quad <c_z, \phi_z, f_z> \tag{7.14}$$

Composition of these fragments involves both leftmost substitution over the c-structures and unification over the f-structures.

The production of LFG-DOP fragments via *discard* involves generating all possible f-structure fragments for each fragment produced via *root* and *frontier* while keeping c-structure and ϕ-links constant. Therefore, the back-off relation is defined in terms of f-structure unification rather than fragment composition. For discard-generated fragments, a directed edge points from a fragment $<c, \phi, f>$ to a pair of fragments $<<c, \phi, f_y>,<c, \phi, f_z>>$ if f-structures f_y and f_z unify to give f and $f \neq f_y \neq f_z$; this is expressed in equation (7.15):

$$<c, \phi, f> \quad \geq_{bkf} \quad <c, \phi, f_y \cup f_z> \tag{7.15}$$

The probability of the derivation $<c, \phi, f_y \cup f_z>$ is given in (7.16):

$$\begin{aligned} & P(<c, \phi, f_y \cup f_z> | R_c) && = \\ & P(c, \phi | R_c) P(\{f_y \cup f_z\} | c, \phi) && = \\ & P(c, \phi | R_c) P(f_y | c, \phi) P(f_z | c, \phi, f_y) && \approx \\ & P(c, \phi | R_c) P(f_y | c, \phi) P(f_z | c, \phi) \end{aligned} \tag{7.16}$$

Thus, the derivation $<c, \phi, f_y \cup f_z>$ embodies an independence assumption realised by the approximation $P(f_z|c, \phi, f_y) \approx P(f_z|c, \phi)$. This approximation constitutes a back-off, hence the derivation $<c, \phi, f_y \cup f_z>$ is said to be a back-off of fragment $<c, \phi, f>$. Here, the back-off relationship is used to organise the set of discard-generated fragments into a back-off graph and this graph can then be used to re-estimate fragment probabilities by transferring probability mass from more specific fragments to their back-offs in a stepwise fashion.

7.4.2 Applying discard in practice

Empirical experience makes clear the need to address the problem of estimating the probabilities of discard-generated fragments as, in practice, these fragments occupy a disproportionately large amount of probability mass. The 'discount RF' parameter estimation method of Bod and Kaplan (2003) successfully masks this problem by treating discard-generated LFG-DOP fragments as second-rate fragments to be used only when no parse can be produced without them. However, the difficulty in estimating the probabilities of discard-generated fragments stems from the bias introduced by using the relative frequency estimator and the discount RF method does not address this issue. In contrast, the back-off estimation method structures and estimates these parameters such that a kind of 'soft' probabilistic back-off is realised and non-discard fragments are preferred naturally (Hearne and Sima'an, 2003).

Allowing the probability model to naturally select the most specific parse by assigning little probability mass to analyses derived using discard-generated fragments is attractive in theory. Such large numbers of discard-generated fragments are possible, however, that, in practice, we are unlikely to have the luxury of including all fragments in the parse space for each string and simply letting the probabilities decide. Furthermore, experiments show that where a parse can be produced without using discard-generated fragments, the inclusion of such fragments does not significantly improve parse accuracy (Bod and Kaplan, 2003). This suggests that discard-generated fragments should only be included in the parse space where it is not possible to generate at least one parse without them. Way (2001) also observes that discard should be used to derive fragments only where absolutely

necessary. He suggests that there must be a countable number of cases – such as subject-verb agreement, relative clause agreement and movement phenomena – in which unification fails and discard should be applied. He proposes that the process of generating and including discard-generated fragments should be triggered by the *occurrence* of unification failure and controlled by the *type* of failure that occurred.

We suggest that the ordering of the fragment base specified by the back-off graph could also be used to motivate the phased addition of discard-generated fragments to the parse space (Hearne and Sima'an, 2003). For example, fragments can be added to the parse space layer by layer, starting with the most specific and working towards the least specific. As soon as at least one parse can be produced, no more fragments are introduced and the most probable parse is determined, thus favouring parses with more complete f-structures. This approach accounts for the fact that, where one or more parses can be produced using discard fragments but none without, these parses can be considered to occupy a spectrum ranging from most specific to least specific, depending on the number of attribute-value pairs discarded from the fragments used to derive them. Other configurations can also be envisaged. For example, following from the proposals in Way (2001), the fragment space could be partitioned based on the *type* (as opposed to number) of simple attribute-value pairs which have been discarded from each fragment.

7.5 Implementing LFG-DOP

The LFG-DOP parser implementation described by Bod and Kaplan (2003) uses the rewrite rule technique of Bod (1998) described in section 2.4.1 to compute the LFG-DOP parse space. Here, we discuss the application of the Tree-DOP algorithms of Sima'an (1995a, 1999) and Goodman (1996a, 1998, 2003) to the LFG-DOP model. Building the LFG-DOP parse space is, in fact, analogous to building the Tree-DOT translation space and, correspondingly, adaptation of Sima'an's algorithm is straightforward. Furthermore, the similarity of the LFG-DOP fragments to the Tree-DOT fragments also allows direct application of the compact fragment representation methodology described in section 3.1.2. Use of Goodman's algorithm is, however, problematic, as is the computation of exact sampling probabilities during disambiguation.

7.5.1 Computing the LFG-DOP parse space

PCFG-reduction for LFG-DOP

Applying the PCFG-reduction method of (Goodman, 1996a, 1998, 2003) to the implementation of LFG-DOP is problematic, not just because the PCFG-reduction must characterise fragments which comprise f-structures as well as phrase-structure trees, but because it is not clear how to apply the Maximum Constituents Parse disambiguation strategy (discussed in section 2.5.6) to select LFG-DOP output. Thus, we do not currently see a way to adapt this parsing methodology for computation of the LFG-DOP parse space.

Two-phase analysis for LFG-DOP

Implementing any of the LFG-DOP models M_1, M_2 and M_3 requires that the parse space be constructed on the basis of the category-matching condition only as the other LFG wellformedness conditions are verified either during or after disambiguation. Consequently, Sima'an (1995a, 1999)'s two-phase analysis methodology can be applied to the construction of the LFG-DOP parse space in a straightforward manner. Firstly, the parse space is approximated using the CFG underlying the c-structures present in the fragment base. Then, correspondences between these rules and the c-structure fragments in which they occur allows for the retrieval of all c-structure fragments – and, correspondingly, f-structure fragments – relevant to the parse space. Disambiguation techniques can then be applied to this LFG-DOP parse space as described in section 7.5.3.

7.5.2 Compact LFG-DOP fragment representations

As for Tree-DOP, using the two-phase algorithm to compute the LFG-DOP parse space for each input string requires only an indication as to which fragments each underlying CFG rule appears in. Therefore, we need to store only the LFG representations themselves, and can establish the fragment set on the fly. This is accomplished in exactly the same manner as for Tree-DOP as described in section 3.1.2 by introducing annotations at each c-structure node. Establishing LFG-DOP fragment frequencies is also straightforward: two intermediate representations I_x and I_y encode duplicate LFG-DOP fragments if connected

portions of those c-structures which start at their root nodes are identical *and* those connected portions specify identical f-structure constraints.

7.5.3 Monte Carlo sampling for LFG-DOP

Firstly, we look at application of the exact sampling technique of Chappelier and Rajman (2003) to LFG-DOP disambiguation. We would like to be able to use this methodology because it allows precise control over the number of samples to be taken. However, we find that computation of the exact probability of sampling each LFG-DOP fragment is not possible due to the global effect of fragment composition on every sub-derivation. We then turn our attention to the computation of re-scored sampling probabilities, which successfully avoids this problem.

Computing exact sampling probabilities for LFG-DOP fragments

Recall that, as given in equation (2.27) in section 2.5.1, computing the exact probability of sampling Tree-DOP fragment f from parse space position [i][j] requires multiplication of the Tree-DOP probability of f, $P_{DOP}(f)$, with the total probability mass available at each of its substitution sites.

n ... $(j+1)$	...	...	...
j	...	... A (B$_{[i][k]}$, C$_{[k+1][j]}$) ...	...
$(j-1)$ ... (1)	...	...	...
	$1...(i-1)$	i	$(i+1)...n$

(7.17)

For example, the probability of sampling the fragment with root node A from the parse space given in (7.17) is its DOP probability by the total probability mass at $B_{[i][k]}$ and by the total probability mass at $C_{[k+1][j]}$; this probability is then divided by the sum over the probabilities of all fragments with root node A at chart position [i][j]. Thus, assuming a

sufficiently large sample set, if the Tree-DOP probability of parse P given the input string is $\frac{n_p}{N_p}$ (where N_p is the sum over the probability mass assigned to each valid parse for the input string) and the frequency of parse P in the set of sampled parses is $\frac{n_s}{N_s}$ (where N_s is the total number of samples taken) then $\frac{n_p}{N_p} = \frac{n_s}{N_s}$.

If we wish also to sample LFG-DOP derivations such that the relative frequency of each LFG-DOP parse in the sampled set equals the LFG-DOP probability of that parse (conditioned on the total probability mass of all valid parses for the input string), then we must compute the exact probability of sampling each LFG-DOP fragment as is done for Tree-DOP. However, we find that computing these probabilities for LFG-DOP fragments is simply not possible due to the global effect of the f-structure constraints imposed by each possible composition. Consider, again, the parse space given in (7.17) but, this time, assume that all fragments are associated with constraints encoded in ϕ-linked f-structures. In order to establish exactly the probability of selecting the given fragment with root node A at position [i][j], we must multiply its DOP probability by the total probability mass assigned to the set of fragments competing for selection at each of its substitution sites. However, we cannot know exactly which fragments are eligible for composition with substitution site $C_{[k+1][j]}$ until a fragment has been composed at substitution site $B_{[i][k]}$ *and* there are no remaining open substitution sites dominated by $B_{[i][k]}$. Thus, we cannot accurately compute the probability of selecting any fragment with more than one open substitution site at any chart position.

Computing rescored sampling probabilities for LFG-DOP fragments

The single greatest advantage in sampling using the exact sampling probabilities of Chappelier and Rajman (2003) is the control it gives over the number of samples taken when disambiguating each parse space. Because the relative frequencies of the sampled parses correspond exactly to their DOP probabilities, we can check after every sample is taken whether or not enough samples have been seen to be sure that the most frequent parse is also the most probable one. Nevertheless, DOP fragments can also be sampled such that, while the relative frequencies of sampled parses in the sample set do not correspond to their DOP probabilities, rescoring factors can be applied to these relative frequencies such

that the correct distribution is induced. Here, we examine the calculation of LFG-DOP fragment probabilities and rescoring factors according to the method given by Hoogweg (2000) for Tree-DOP which was described in section 2.5.1.

Using Hoogweg (2000)'s methodology, the probability of sampling any given fragment f is the DOP probability of f conditioned on the sum of the DOP probabilities of all fragments in its competition set. This competition set is established on the basis of the sub-derivation seen so far. Thus, this method does not take into consideration the sampling probability mass available at each of f's substitution sites and, consequently, does not face the difficulties of computing exact sampling probabilities which we identified above. Furthermore, the rescoring factor which must be applied to each sampling frequency – calculated for each derivation as the product of the competition set probability masses for each fragment involved in building that derivation – can be computed without difficulty as each derivation is sampled. Thus, this method can be applied in a straightforward manner when performing LFG-DOP sampling.

7.6 Summary

In the first two sections of this chapter, we described the work which has been carried out to date on the LFG-DOP parsing model, including the available implementation details and the results achieved using this system. We then went on to highlight two specific issues which arise as a result of how the fragment extraction process is defined – namely, the handling of circular and re-entrant structures, and the retention of non-governable attributes for which there is no c-structure evidence – and addressed these issues by proposing an alternative definition for fragment extraction. As LFG-DOP also suffers from the bias towards larger parses exhibited by the Tree-DOP model when fragment probabilities are defined using the relative frequency estimator, we discussed an alternative method for parameter estimation which extends the back-off re-estimation methodology described for Tree-DOP. We highlighted how this method not only provides a possible solution to the model bias but also provides a natural solution to the difficulty of incorporating robustness into the LFG-DOP model. (We note, however, that this method of assigning probabilities to fragments does not address the issue of 'leaked' probability mass raised by Abney (1997)

discussed in section 7.1.4.) Finally, we looked at the application of efficient algorithms developed for Tree-DOP to parsing with the LFG-DOP model. We described why Goodman (1996a, 1998, 2003)'s PCFG-reduction method for computing the parse space appears unsuitable LFG-DOP parsing and also outlined the difficulties of calculating exact sampling probabilities over the LFG-DOP parse space. However, we showed that Sima'an (1995a, 1999)'s two-phase analysis method can be used to construct the LFG-DOP parse space in a straightforward manner, and that re-scored sampling probabilities can be computed over this parse space such that the sample set distribution corresponds to the distribution of parses for the input string according to the LFG-DOP model.

Chapter 8

A richer DOT model: LFG-DOT

The motivations for merging the Tree-DOT model of translation with the LFG-DOP parsing model are clear. The constraints expressed in the LFG f-structures allow a more linguistically-precise description of how translations should be formed than the simple phrase-structure fragments used in Tree-DOT. Furthermore, although these functional descriptions embody a greater degree of complexity than phrase-structure trees, the translation process is still driven by the c-structures with which they are associated. This means that the algorithms adapted to Tree-DOT and LFG-DOP can also be applied to fragmentation, analysis and disambiguation for the LFG-DOT model.

In section 8.1 of this chapter, we present the LFG-DOT models of translation described by Way (2001) and in section 8.2, we present an alternative LFG-DOT translation model. We outline the application of the implementation methodologies described for Tree-DOT and LFG-DOP to the LFG-DOT model in section 8.3. We then go on in section 8.4 to discuss the relationship between translational equivalence and limited compositionality for Tree-DOT and LFG-DOT, and show how they differ depending on the representations assumed to underlie the model. Finally, we hypothesise as to how the feature sets for LFG-DOT fragments might be pruned in section 8.5.

8.1 The LFG-DOT models of Way (2001)

In this section, we describe the four different LFG-DOT models proposed by Way (2001). The first three models vary in terms of how the translation relation is stated, while the

fourth augments the third through an extension to the discard operation.

8.1.1 LFG-DOT Model 1

LFG-DOT Model 1 is a simple, linear model. The source-language input string is parsed using the LFG-DOP model which yields an LFG representation for that string, i.e. a context-free phrase-structure tree, an attribute-value matrix and the links between them. The phrase-structure tree is discarded and the attribute-value matrix, or f-structure, from this representation is passed into a rule-based transfer component which yields a target-language f-structure. This f-structure is then passed to a target-language LFG-DOP generation model which outputs the most likely string. This architecture is summarised in (8.1).

(8.1)

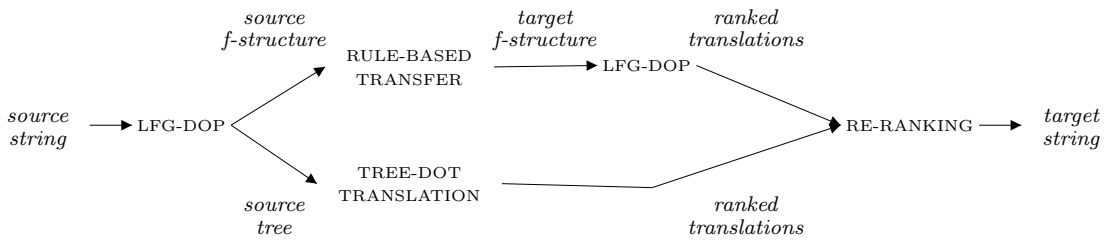
This model essentially corresponds to a typical rule-based system architecture, despite the fact that the analysis and generation components are data-driven. It does not require that any correspondences be drawn between the source-language representations assumed by the parser and the target-language representations assumed by the generation component. Thus, the characteristics of the Tree-DOT model are not inherited by this model. Furthermore, Way (2001) observes that the probability model associated with this architecture embodies the undesirable assumption that the target string is generated from the target-language f-structure independently of the source string. This model is given in equation (8.2).

$$P(t|s) = \sum_{R_{s,t}} P(R_s|s)\ P(R_t|R_s)\ P(t|R_t) \tag{8.2}$$

While the probability of the target f-structure generated clearly depends on the source string as this source string is contained within the source representation, the probability of the target string generated by the monolingual language model is not influenced by the source string and its representation.

8.1.2 LFG-DOT Model 2

LFG-DOT Model 2 is more complex than Model 1 in that both rule-based and data-driven translation components are employed in parallel. In this model, translational correspondences are assumed between both the source and target c-structures and (independently of the c-structure links) between the source and target f-structures. The linked c-structures are used in a Tree-DOT model while the f-structure links yield the set of rules which comprise the transfer component. As for Model 1, the source-language input string is parsed using an LFG-DOP parser, resulting in an LFG representation for that string. In this model, however, both the tree and the f-structure yielded by the parser are used in the translation process: the tree is input to the Tree-DOT translation model which yields a set of ranked translations and, simultaneously, the f-structure is input to the rule-based transfer component which yields a target-language f-structure. The target f-structure is then passed to a target-language LFG-DOP generation model which outputs a set of ranked translations. The sets of ranked translations yielded by the Tree-DOT component and linear application of the rule-based and LFG-DOP components are then merged and the most probable translation determined. This architecture is summarised in (8.3).



(8.3)

The probability model given in (Way, 2001) for LFG-DOT Model 2, given in (8.4) below, corresponds to the intuition that the translation probability is the sum over the probabilities of all paired bilingual LFG representations that yield both the source and target strings.

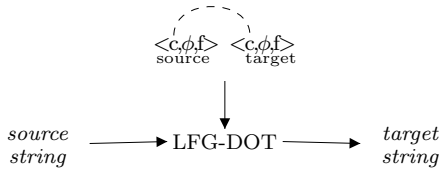
$$P(t|s) = \sum_{R_{s,t}} P(R_{s,t}) \tag{8.4}$$

However, it is not clear how these probabilities should be computed given the model architecture. As the target c-structure and target f-structure representations are never combined to form a single representation and, furthermore, the two need not necessarily

be in correspondence, it is difficult to see how overall representation probabilities can be established. In addition, no methodology is given to combine the ranked sets of translations output by tree translation and f-structure translation in order to determine which translation is the most probable overall.

8.1.3 LFG-DOT Model 3

LFG-DOT Model 3 assumes a bilingual LFG-annotated corpus with translational links between source and target c-structures only. Way (2001) describes this model as an extension of the Tree-DOT model whereby the f-structures are used for monolingual filtering and play little part in the translation process itself. Essentially, the f-structures exert monolingual constraints over the source representations and target representations which can be formed and, therefore, ensure that the translations generated are grammatical. Importantly, these constraints can be relaxed using discard so that input which is ungrammatical with respect to the corpus can also be handled. The architecture of Model 3 is summarised in (8.5).



(8.5)

Clearly, this model improves over the Tree-DOT model as ungrammatical target-language strings are filtered out. However, Way (2001) also shows – using examples of complex translation cases and combinations of exceptions – that this architecture improves over LFG-DOT Model 1, where translation takes place over f-structure rather than c-structure. He indicates that this is due to the arbitrary size of the c-structure fragments used in the translation model in combination with the additional linguistic information available in the f-structures which allows ungrammatical translations to be ruled out.

The probability model given in (Way, 2001) for LFG-DOT Model 3, given in (8.6) below, again corresponds to the intuition that the translation probability is the sum over the probabilities of all paired bilingual LFG representations that yield both the source and

target strings.

$$P(t|s) = \sum_{R_{s,t}} P(R_{s,t}) \tag{8.6}$$

Here, however, he specifies the following components:

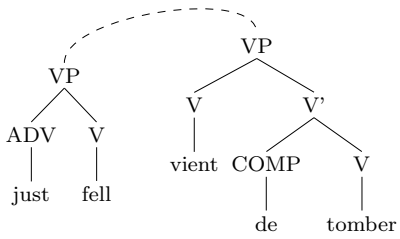
1. $P(R_s)$: the probability of the source-string LFG representations, and

2. $P(R_t|R_s)$: the Tree-DOT transfer probability.

He states that the probability of the source representation $P(R_s)$ is calculated according to the source-language LFG-DOP probability model. The Tree-DOT translation probability $P(R_t|R_s)$, on the other hand, is estimated by dividing the probability of the linked source and target LFG representations by $P(R_s)$. As the target string is read from the leaf nodes of the target c-structure, no further calculation is required. However, it is not clear how these probability calculations should proceed, given the model architecture.

8.1.4 LFG-DOT Model 4

Way (2001) observes that Tree-DOT fragments suffer from 'limited compositionality', meaning that some of them can be used to build derivations only under very limited circumstances where no further generalisation is possible. For example, the fact that the V nodes dominating the terminals *fell* and *tomber* in the linked subtree pair in example (8.7) are not linked means that no fragment can be extracted which captures the generalisation *just* $V_s \longrightarrow$ *vient de* V_t.

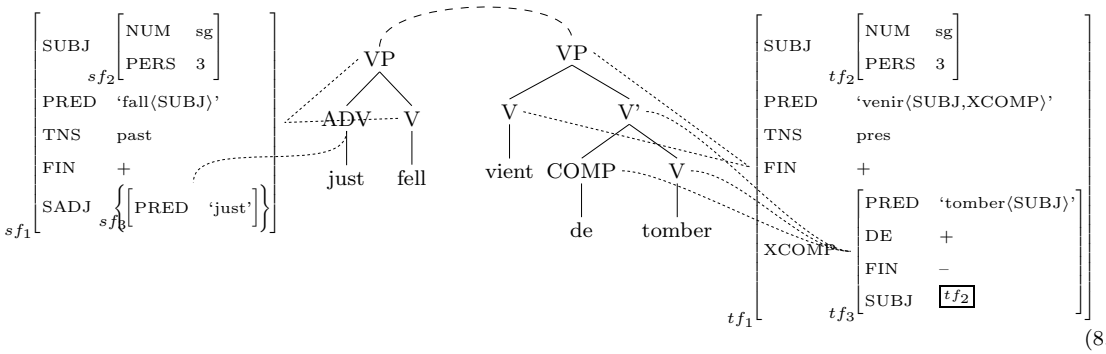


(8.7)

This link does not exist because *fell* and *tomber* are not considered translationally equivalent due to the fact that the verb form *fell* is in the past tense whereas the verb form *tomber* is in the infinitive.

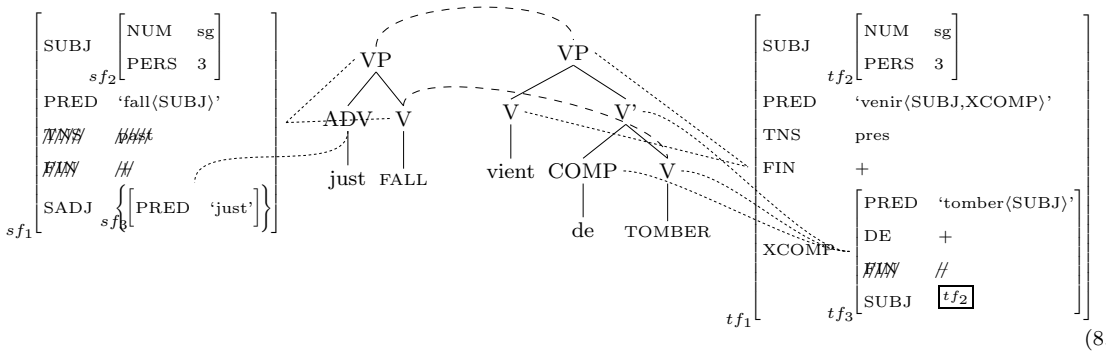
LFG-DOT Model 3 inherits this characteristic from Tree-DOT, as the links between the assumed representations correspond exactly to Tree-DOT links. In LFG-DOT Model

4, however, Way (2001) proposes to extend the discard operation in order to address this issue. In example (8.8), we show the LFG-DOT fragment corresponding to the Tree-DOT fragment given in (8.7). Again, no link exists between the subtrees $V \longrightarrow fell$ and $V \longrightarrow tomber$. Here, however, the features contained in the f-structure overtly indicate why these two nodes are not linked by explicitly stating their tense features.



(8.8)

Recall that discard is used to delete attribute-value pairs from the f-structure whose values are not ϕ-linked to remaining nodes in the c-structure and are not surface forms corresponding to remaining c-structure terminals. Way (*op. cit*) suggests that if the tense features of these verbs are deleted using discard and their c-structure realisations lemmatised then these verbs can be linked, as shown in example (8.9) below. From this fragment, we can establish the required generalisation *just* $V_s \longrightarrow$ *vient de* V_t.



(8.9)

Way (*op. cit*) suggests that verb forms only should be subject to lemmatisation. Furthermore, he suggests that no discard-generated fragments should be introduced into the translation space until it has been established that no translation can be produced without them.

8.2 A new LFG-DOT model

In this section, we provide an alternative LFG-DOT model which corresponds exactly to the architecture of the LFG-DOT Model 3 of (Way, 2001) but differs in how the fragments are extracted and how the probabilities are computed.

As always when specifying a data-oriented model of natural language, we specify the underlying representations assumed, how fragments are to be extracted from those representations, how those fragments are to be recombined to form output for previously-unseen input strings and, finally, the probability model to be used to rank the output. Here, our specification essentially involves merging the sets of specifications described for Tree-DOT in section 4.2.1 and LFG-DOP in section 7.1.

8.2.1 Representations

The representations assumed by the LFG-DOT model comprise pairs of LFG $<c, \phi, f>$ triples which have been aligned at sub-structural level by placing links denoting translational equivalence between source and target c-structure nodes. Thus, they resemble Tree-DOT representations in that they comprise linked source and target phrase-structure trees but they also resemble LFG-DOP representations in that each of these source and target phrase-structure trees is linked to an attribute-value matrix. An example LFG-DOT representation is given in Figure 8.1 – the dotted lines represent ϕ-links associating c-structure nodes with f-structure units and the dashed lines represent translational links denoting source and target sub-structures which are translations of each other. If we take away the dotted lines and attribute-value matrices, we are left with Tree-DOT representations and, conversely, if we take away the dashed lines we are left with monolingual LFG-DOP representations.

8.2.2 Fragmentation

In order to define how fragments should be extracted from LFG-DOT representations, we merge the fragment extraction definitions given for Tree-DOT and LFG-DOP. In other words, fragmentation respects both the links denoting translational equivalence and the

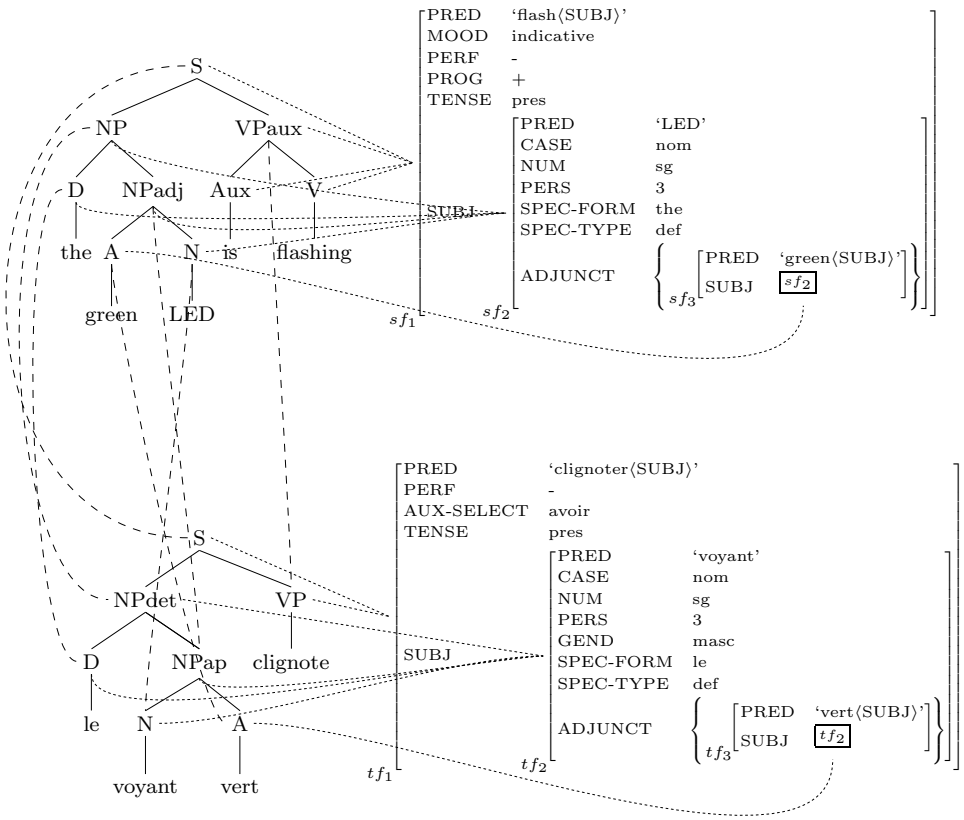


Figure 8.1: An example aligned bilingual LFG representation where the dotted lines show LFG ϕ-links while the dashed lines denote translational equivalences.

links denoting c-structure and f-structure correspondences. Thus, fragmentation proceeds as follows.[1]

1. Extract a c-structure fragment pair from an LFG-DOT representation using the root and frontier operations as for Tree-DOT – that is, select a pair of linked nodes to be root, deleting all nodes not dominated by them, and then select a (possibly empty) set of paired linked nodes to be frontiers and delete all that they dominate – but associate with both of these c-structure subtrees the full f-structures with which they are associated in the original representation.

2. For the source and target representations in turn, delete all f-structure units (and the attributes with which they are associated) which are not ϕ-linked from one or more remaining c-structure nodes *unless* that unit is the value of an attribute subcategorised for by a PRED value whose corresponding terminal is dominated by the current fragment root node in the original representation.

   (a) Where we have floating f-structure units – i.e. a fragment is associated with f-structure units f_x and f_y such that f_x does not contain f_y and f_y does not contain f_x – then we also retain the minimal f-structure unit which contains them both. By minimal unit we mean the unit comprising the attribute with value f_x and the (nested sequence of minimal units containing) attribute with value f_y.

3. For the source and target representations in turn, delete all semantic forms (including PRED attributes and their values) not associated with one of the remaining c-structure terminals.

Figure 8.2 gives examples of LFG-DOT fragments extracted from the LFG-DOT representation given in Figure 8.1. Consider, for example, fragment (a) in Figure 8.2. This fragment was extracted by selecting the linked c-structure node pair <VPaux,VP> to be the root node pair and deleting all c-structure nodes not dominated by this node pair. As this root node pair does not dominate any further linked node pairs, the frontier operation

[1] We establish the attribute-value pairs to be retained for each fragment according to the new definition for LFG-DOP fragments given in section 7.3 on page 199.

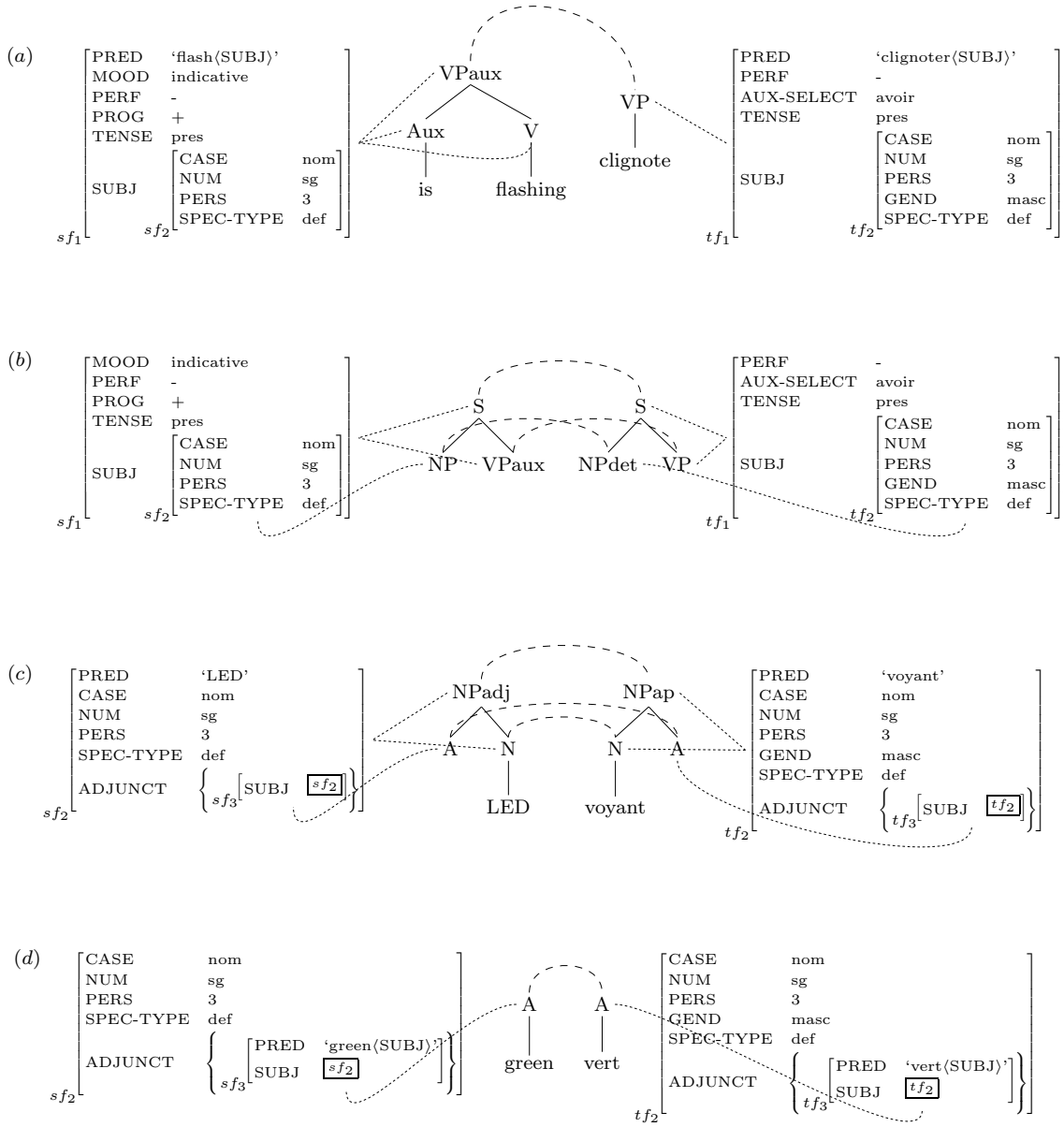


Figure 8.2: Example LFG-DOT fragments extracted from the representation in Figure 8.1.

must select the empty set. This c-structure fragmentation procedure thus corresponds to fragmentation clause (1) above. Selection of the f-structure units corresponding to the source and target tree-structures proceeds according to clause (2). As units sf_1 and tf_1 are linked to c-structure nodes, all simple attribute-value pairs which are members of these units are retained. Units sf_2 and tf_2 are not linked to c-structure nodes, but as their attributes are subcategorised for by the source and target PRED values 'flash<SUBJ>' and 'clignoter<SUBJ>', these units are also retained. Units sf_3 and tf_3, however, are neither linked to c-structure nodes nor subcategorised for, and so these units, along with their attributes, are deleted. Finally, following clause (3) above, semantic forms corresponding to deleted c-structure terminals are deleted. Thus, fragment (*a*) expresses the fact that if *is flashing* is to be translated as *clignote* then, for example, both source and target sentences are in the present tense and must have nominative subjects. Furthermore, as the value of the non-governable ADJUNCT attribute is no longer linked to a c-structure node in either the source or target sides of the representation, it is not a requirement that these subjects be modified; this is also the case for fragment (*b*).

Fragment (*c*) in Figure 8.2 was extracted by selecting the linked c-structure node pair <NPadj,NPap> to be the root node pair and deleting all c-structure nodes not dominated by this pair. The frontier operation then selected a set of linked node pairs comprising only the pair <A,A> and deleted the subtrees dominated by these nodes. Again, this c-structure fragmentation procedure thus corresponds to (1) above and, again, selection of the f-structure units corresponding to the source and target tree-structures corresponds to (2). As f-structure units sf_2, tf_2, sf_3 and tf_3 are all linked to c-structure nodes, they are automatically retained. In contrast, the outer f-structure units sf_1 and tf_1 are neither linked to nor subcategorised for, and so all other attribute-value pairs in these units are themselves deleted. Finally, following (3) above, remaining semantic forms corresponding to deleted c-structure terminals are deleted. Thus, fragment (*c*) states not only that the noun *LED* translates as the noun *voyant* when both are modified by an adjective, but also that those source and target adjectives must be in agreement with the nouns in question. The converse translational relationship is stated in fragment (*d*), where we see that the f-structures associated with the source and target fragments specify the types of nouns

each adjective can modify. For example, the target side of the fragment specifies that the modified noun must be a singular masculine noun. This distinguishes the surface form specified in the c-structure from the other possible surface realisations of the lexeme vert, i.e. the masculine plural form *verts*, the feminine singular form *verte* and the feminine plural form *vertes*.

8.2.3 Composition

The definition of LFG-DOT fragment composition merges the composition procedures for Tree-DOT and LFG-DOP. As for Tree-DOT, each unlinked c-structure frontier node is a terminal symbol and each linked c-structure frontier node is a syntactic category. In composition terms, each pair of linked frontier nodes constitutes an open substitution site, and fragments whose linked source and target root nodes are of the same syntactic category as the linked source and target substitution site categories can be considered for substitution at these frontiers. However, as for LFG-DOP, the f-structures associated with each c-structure being substituted must unify with the f-structures of the current sub-derivation. For LFG-DOT, this requirement applies to both the source and target fragments – that is, the source f-structure of the fragment being substituted must unify with the current source sub-derivation f-structure and the target f-structure of the fragment being substituted must unify with the current target sub-derivation f-structure.

Consider, for example, the topmost fragment in Figure 8.3, where the leftmost open source substitution site and its target linked counterpart are of category <NP,NPdet>. All fragments whose linked c-structure root node categories are also <NP,NPdet> are considered for substitution. However, only fragments whose source and target f-structures unify successfully with the f-structures of this fragment – such as the one directly beneath it in Figure 8.3 – can actually be selected for substitution. Note that, although our definitions of LFG-DOT and LFG-DOP fragments is less restrictive than those of Bod and Kaplan (2003) and Way (2001) in that we do not specify an adjunct to the subject, the constraints specified are nevertheless quite restrictive. For example, the fragment with root node <S,S> specifies that any subject noun phrase combined with it must be third person singular and, in the case of the French noun phrase, of masculine gender.

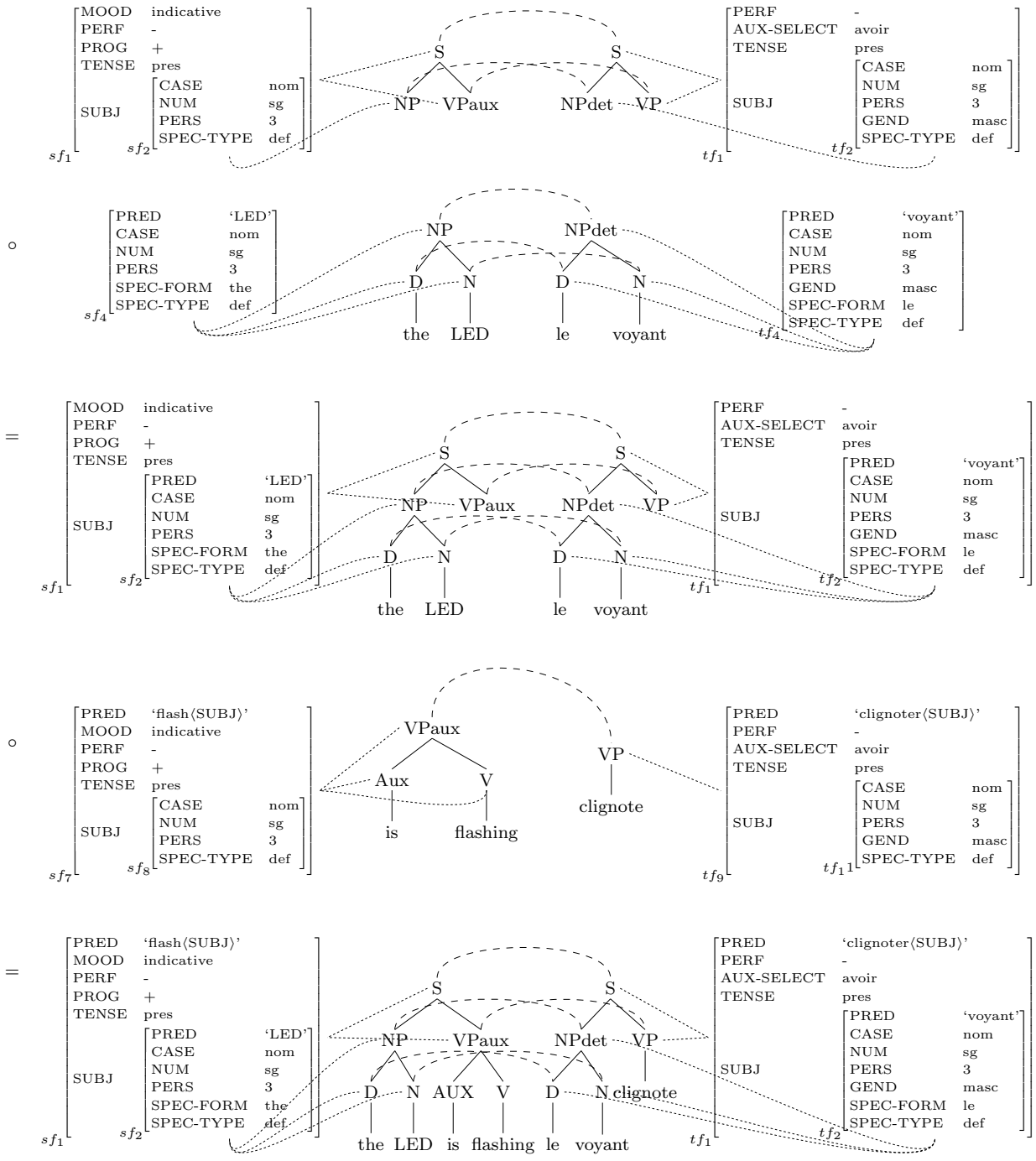
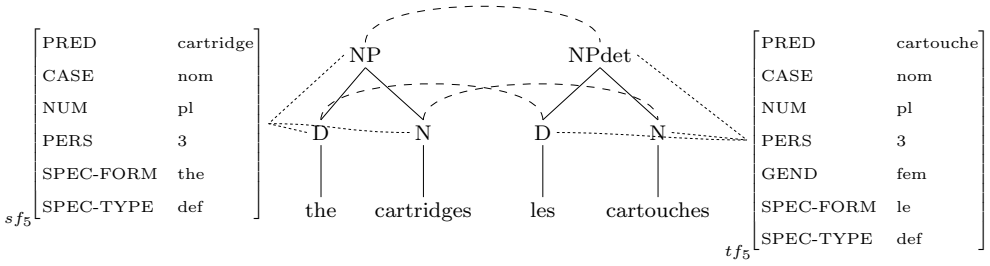


Figure 8.3: Example derivation using LFG-DOT fragments and the LFG-DOT composition operation.

Thus, although the fragment shown in example (8.10) below meets the category-matching criterion, unification with this <S,S>–rooted fragment fails because both the source and target noun phrases specify number plural and, in addition, the target noun phrase specifies gender feminine.



(8.10)

Note that the labels identifying f-structure units do not play any part in the composition process. Rather, it is the ϕ-links that indicate at which units unification is to take place. For example, source f-structure unification when composing the top two fragments in Figure 8.3 is dictated by the ϕ-link indicating that the substitution site NP corresponds to f-structure unit sf_2. Therefore, the source f-structure of the fragment substituted at node NP must compose with unit sf_2, irrespective of its label.

As for all previous data-oriented models presented, a derivation is complete only when no open substitution sites remain. Thus, the sub-derivation yielded by the composition of the top two fragments in Figure 8.3 is incomplete as linked substitution site <VPaux,VP> remains. Composition of the fragment representing the paired strings *is flashing* and *clignote* with this sub-derivation does, however, yield a complete derivation.

8.2.4 The probability model

As for Tree-DOT, the LFG-DOT probability model requires that we sum over the probabilities of all valid derivations yielding a given translation in order to determine which translation is the most probable. Thus – in theory, at least – we generate all possible bilingual representations for the input string and establish the target string probabilities accordingly. As for LFG-DOP, however, not all derivations are valid as some may be in violation of the LFG uniqueness, completeness and coherence well-formedness conditions. Thus, we must remove all invalid derivations from the probability space and condition

our final probabilities on the total *valid* probability mass only. Although LFG-DOP models M_1 and M_2 can also be used, the probability model we describe here corresponds to LFG-DOP Model M_3.

Again, building an LFG-DOT derivation can be viewed as a top-down stochastic branching process. A fragment whose root node pair corresponds to the start category pair is selected at random to start the derivation. Further fragments are successively chosen to combine with the leftmost open substitution site of the derivation; these fragments are chosen at random from the set of fragments competing for selection at each substitution site. Thus, the competition probability of selecting a fragment at random to participate in a derivation is the likelihood with which it is drawn from the competition set (CS), i.e. its probability over the total probability mass assigned to the CS as given in equation (8.11).

$$CP(f) = \frac{P(f)}{\sum_{f' \in CS} P(f')} \tag{8.11}$$

As we choose to enforce the LFG uniqueness and coherence conditions during this sampling process, the set of fragments eligible for substitution at each site must have the appropriate root node pair and be unifiable with both the source and target f-structures without violating either the uniqueness or coherence conditions. This specification of the competition set is given in (8.12).

$$CS = \{f : root(f) = LSS(D_{i-1}) \wedge unique(D_{i-1} \circ f) \wedge coherent(D_{i-1} \circ f)\} \tag{8.12}$$

Of course, as the LFG completeness criterion can only be verified once each sample derivation is completed, some derivations sampled may still be invalid. Therefore, final translation probabilities are calculated only over valid derivations, as given in equation (8.13).

$$P(t|s) = \sum_{R \; yields \; s,t} \frac{P(R)}{\sum_{R' \; is \; valid} P(R')} \tag{8.13}$$

The probabilities of discard-generated fragments can be calculated using the method described for LFG-DOP in section 7.1.4. We note, however, that this method of assigning probabilities to fragments does not address the issue of 'leaked' probability mass raised

by Abney (1997); this is precisely the same problem as discussed for LFG-DOP in section 7.1.4.

The crucial difference between this probability model and the one specified by Way (2001) for LFG-DOT Model 3 is that we assume each bilingual LFG-DOT fragment to be a single unit of information and, correspondingly, each bilingual representation generated for the input string to be a single representation. Thus, we do not break the representation into separate source and target entities for the purposes of probability calculation. In fact, the LFG-DOT model can be viewed as a parser which assigns analyses to the input string such that those analyses happen – due to the underlying representations assumed by the model – to incorporate a target-language string which constitutes a translation of the input. When viewed from this perspective, the only difference between the LFG-DOP and LFG-DOT models is in the probability model: the LFG-DOP probability model maximises representation probability whereas the LFG-DOT probability model maximises target string probability.

8.3 Implementing LFG-DOT

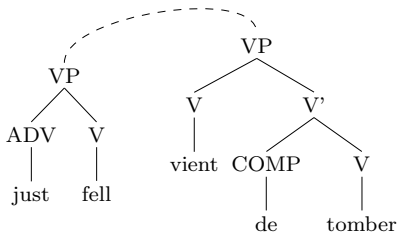
The LFG-DOT model discussed in section 8.2 merges the Tree-DOT and LFG-DOP models. Both the underlying fragments and representations constructed for each input string resemble Tree-DOT representations in that they comprise linked source and target phrase-structure trees and resemble LFG-DOP representations in that each of these source and target phrase-structure trees is linked to an attribute-value matrix. However, the implementation methods applicable to LFG-DOP are fewer than for Tree-DOT. We propose that Sima'an (1999)'s two-phase algorithm be used to build the LFG-DOP parse space as discussed in section 7.5.1. Extension to handle bilingual rather than monolingual fragments is straightforward, as discussed in section 5.2.2. Furthermore, while both exact and re-scored sampling probabilities are computable from the Tree-DOT translation space as discussed in section 5.2.4, only re-scored sampling probabilities are computable from the LFG-DOP parse space, as discussed in section 7.5.3. Thus, implementation of the LFG-DOT model necessitates the combination of two-phase analysis to compute the translation space for the input string and re-scored sampling to select the most probable translation

from that space according to the model. In addition, the compact fragment representations and frequency calculations described for Tree-DOT and LFG-DOP fragments described in sections 5.2.3 and 7.5.2 respectively can be readily applied to LFG-DOT fragments.

8.4 Translational equivalence and limited compositionality

Way (2001)'s LFG-DOT Model 4 extends discard with lemmatisation of c-structure terminals in order to address the issue of limited fragment compositionality inherited by the LFG-DOT model from Tree-DOT. We suggest that, in fact, the differences between the contextual information captured by Tree-DOT and LFG-DOT representations allows us to express different sets of translational dependencies between the same pairs of strings. In order to show how these differences impact on the issue of limited fragment compositionality for the Tree-DOT and LFG-DOT models, we discuss the factors which allow us to decide whether or not two sub-structures are translationally equivalent.

Translational equivalence in Tree-DOT Consider the example Tree-DOT fragment given in example (8.7) and repeated for convenience as example (8.14). We stated that, because no link exists between the V nodes dominating the terminals *fell* and *tomber* in the linked subtree pair, no fragment can be extracted which captures the generalisation *just* $V_s \longrightarrow$ *vient de* V_t. Furthermore, we stated that the reason this link does not exist is because *fell* and *tomber* are not considered translationally equivalent as the verb form *fell* is in the past tense whereas the verb form *tomber* is in the infinitive.

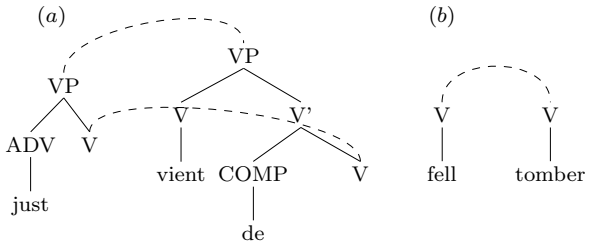


(8.14)

However, this is not entirely accurate: the substrings *fell* and *tomber* clearly *are* translationally equivalent, but only in extremely limited contexts.

If we assume Tree-DOT representations, then our only means of signalling context is through syntactic structure and surrounding terminals. For example, if we choose to link

the nodes dominating *fell* and *tomber* then we can extract the two extra fragments (*a*) and (*b*) shown in (8.15) but the only constraint on the contexts in which these fragments can be used is in terms of syntactic category.

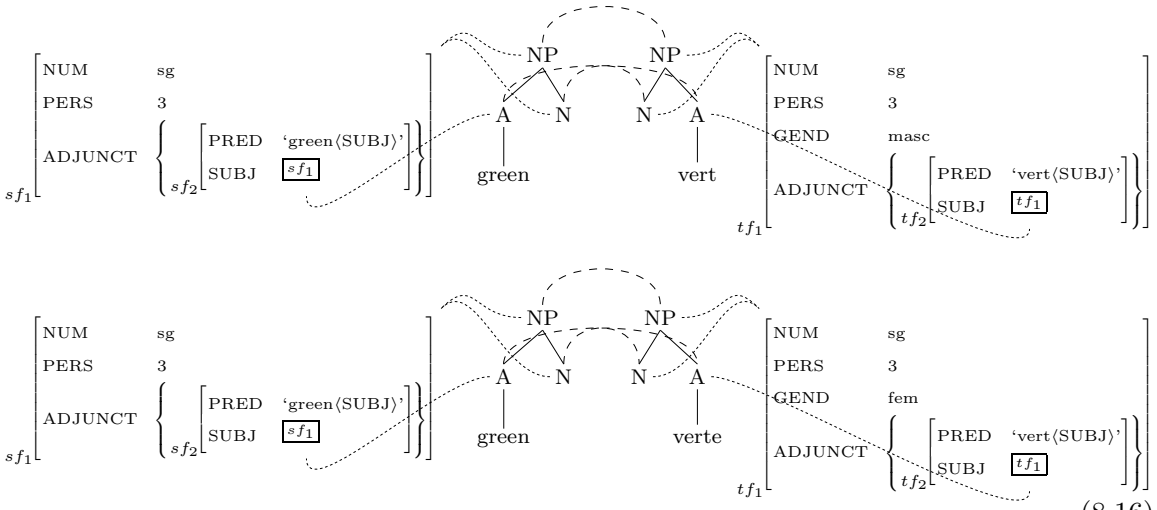


(8.15)

Thus, any pair of linked substrings with root node pair <V,V> can be composed with fragment (*a*) regardless of how they are tensed. For example, if we have a linked, tensed verb pair <*arrived,arriva*> then we can generate the ill-formed translation *vient de arriva*. Similarly, we can substitution fragment (*b*) into any sub-derivation where the leftmost source substitution site and its target linked node are of category V, thereby generating such ill-formed translations as *Martin tomber* for *Martin fell*. Clearly, syntactic category alone does not signal the appropriate context for this particular translation pair. If we are limited to Tree-DOT representations, our only other way of signalling context is through the surrounding terminals and the syntactic structure describing them. Thus, the appropriate context is specified by *not* linking the nodes dominating *fell* and *tomber*, as in the representation given in (8.14) above. In other words, given the representations available to us, this is the most generalised fragment we can extract which still retains enough information to be able to correctly signal the extremely limited context in which *fell* and *tomber* are translationally equivalent.

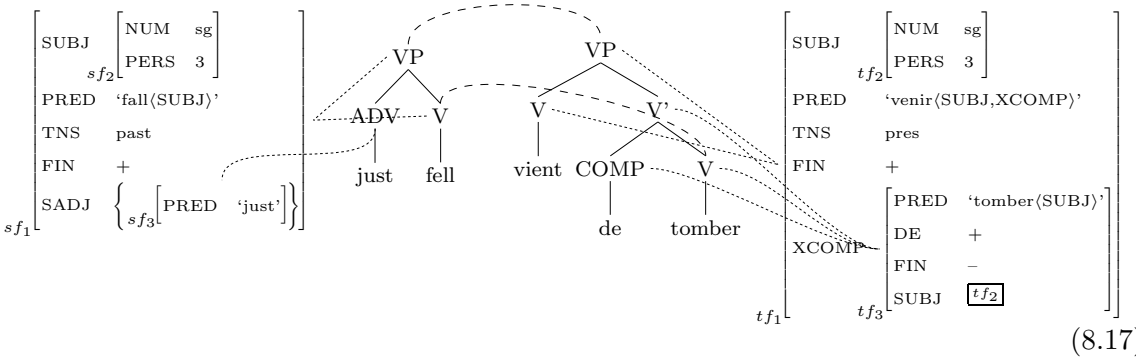
Translational equivalence in LFG-DOT Way (2001) states that the translation relation between LFG-DOT representations is stated only at the level of surface structure and, therefore, the f-structure constraints operate monolingually. We, however, view each LFG-DOT fragment as a single unit of translation information, meaning that the constraints are also translational. (In fact, in both DOT and LFG-DOT we claim that there are no monolingual dependencies modeled at all, rather the only dependencies modeled are translational.) For example, the LFG-DOT fragments given in example (8.16) below

express the translational relationship "*green* $X_s \longrightarrow X_t$ *vert* when $X_s \longrightarrow X_t$ **and** X_t is masculine whereas *green* $X_s \longrightarrow X_t$ *verte* when $X_s \longrightarrow X_t$ and X_t is feminine".



(8.16)

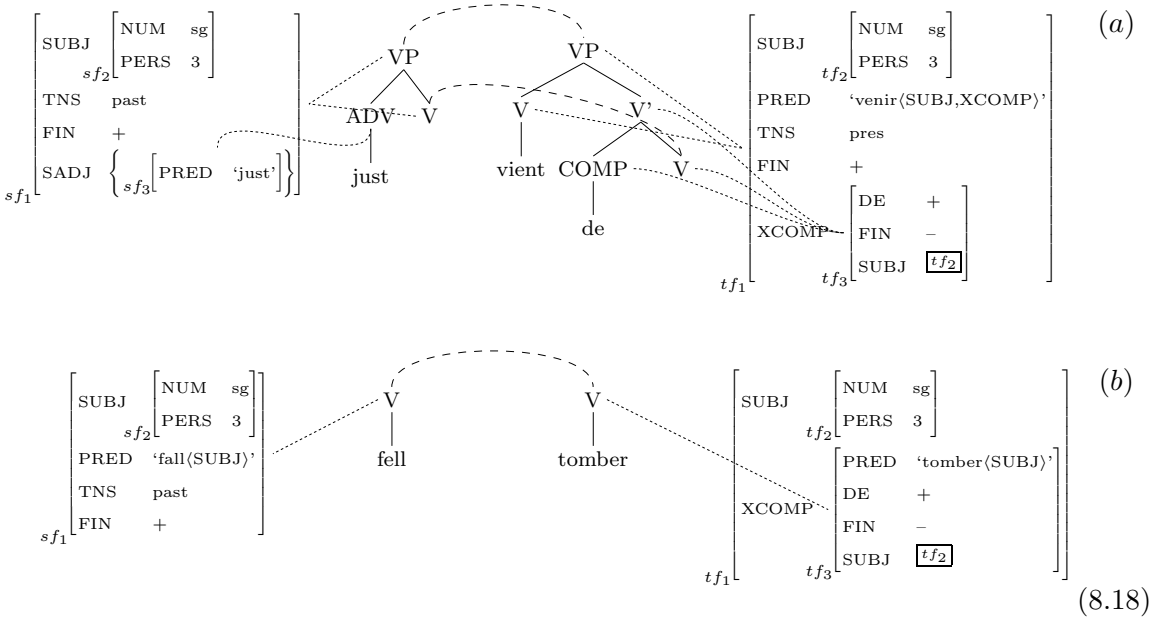
Taking the viewpoint that each LFG-DOT fragment comprises a single unit of translational information means that we now have an additional way of signalling context: as well as using syntactic structure and surrounding terminals, as for Tree-DOT, we can now constrain fragment composition through the functional and grammatical information provided in the f-structures. We now consider the implications of inserting a translational link between the nodes immediately dominating *fell* and *tomber* in the LFG-DOT representation where *just fell* translates as *vient de tomber*, as shown in (8.17).



(8.17)

Here, we see that the f-structure corresponding to the verb *fell* explicitly states that this form is tensed whereas the f-structure corresponding to *tomber* states that this form is in the infinitive. This information is also present in the two extra fragments which can now be extracted from this representation thanks to the presence of the additional translational

link. These two fragments are shown as (a) and (b) in example (8.18).

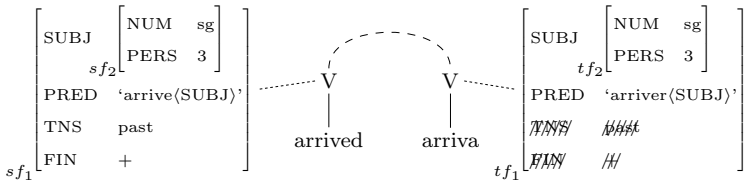


(8.18)

Clearly, these LFG-DOT fragments are far more constrained than their Tree-DOT counterparts. For example, in order for a pair of translationally equivalent source and target verbs to be considered for composition with fragment (a), the source f-structure must agree that the source verb is finite and in the past tense whereas the target f-structure must agree that the verb is the infinitive and functions as an XCOMP. Correspondingly, in order for fragment (b) to be composed with any sub-derivation, the source f-structure of (b) must unify with the source sub-derivation f-structure, meaning that the source sub-derivation must allow a past tense verb. Furthermore, the target sub-derivation must allow a verb in the infinitive to function as an XCOMP in order for it to be unifiable with the target f-structure of fragment (b). Thus, we conclude that the LFG-DOT representations incorporate sufficient contextual information to allow us to link these nodes.

One might wonder how useful such fragments will actually be for real translation tasks, given that, even in very large corpora, we are likely to see few fragments which did not occur in exactly this context originally (in which case less generalised fragments will yield the correct translation) and yet still conform to these very restrictive contexts. However, fragments which do not exactly match the required context can still prove useful through application of discard where no alternative presents itself. For example, the fragment in example (8.19) can compose with fragment (8.18)(a) once the tense features in the target

f-structure have been deleted.



(8.19)

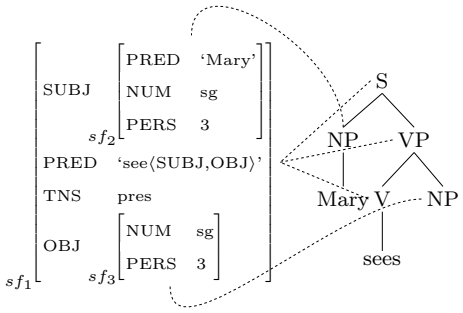
This yields the ungrammatical translation *vient de arriva* for the string *just arrived*. However, as this translation was produced using discard fragments it can be flagged as ungrammatical with respect to the corpus. Furthermore, the substring associated with the deleted features can also be flagged as potentially problematic. Finally, the representation associated with this translation can also be returned; this representation contains the lemma(s) corresponding to the ill-formed substring along with the features needed to generate the correct form.

We note that every case of limited fragment compositionality is not solved by adopting LFG-DOT representations. However, some translation examples embody such complex mappings between source and target strings that overgeneralisation becomes a real concern. Therefore, we conclude that, for such cases, retention of highly-specific fragments is desirable in order to correctly signify context and avoid overgeneralisation.

8.5 Learning features which predict good solutions

In section 7.3, we provided an alternative definition for LFG-DOP (also used for LFG-DOT) as to which f-structure units correspond to c-structure fragments. The difference between this definition and the ones provided by (Bod and Kaplan, 1999, 2003) is that it differentiates between governable and non-governable arguments, only retaining non-governable attributes and their values where those values are ϕ-linked to one or more c-structure fragment nodes. However, once an argument is subcategorised for by a *PRED* value corresponding to a terminal dominated by the fragment root node in the original tree, the f-structure information for this argument is retained. In the fragment given in (8.20), for example, the NUM and PERS features for the object noun phrase are retained

even though the terminal they describe has been deleted.



(8.20)

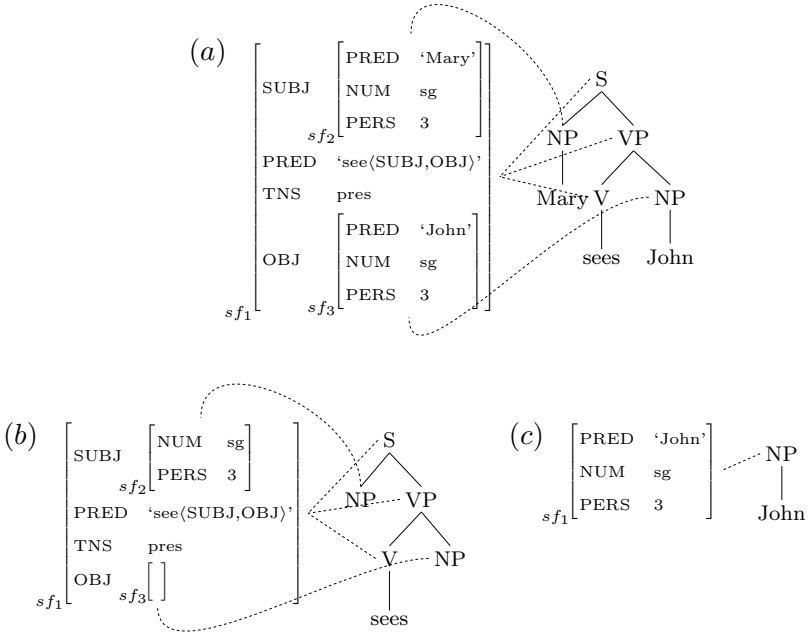
In order to compose a noun phrase of number plural – such as *the girls*, for example – with this fragment, the number feature will have to be discarded from either the object f-structure unit in (8.20) or the plural noun phrase fragment being composed.

The purpose of the discard operation is to allow constraint relaxation where either the input is ill-formed or the evidence in the treebank is insufficient to fully analyse the input. While the input string *Mary sees the girls* is well-formed, if the only fragments in the fragment base relevant to this input string lead to the sub-derivation in (8.20) and, therefore, we must use discard to generate an analysis, then the treebank evidence must be considered insufficient with respect to this input string. We question, however, whether it is really the case that the treebank evidence is insufficient or whether we are, in reality, using discard to compensate for the fact that our f-structure fragments impose unwarranted constraints. In other words, as English does not exhibit object-verb agreement, placing constraints on the features of the expected object in fragment (or sub-derivation) (8.20) seems counter-intuitive.

Given that the LFG-DOP model is not specialised for use in any particular application, we assume that, along with disambiguation, the objective for each input string is to output the most informative parse possible. Suppose, however, that we wanted to use our LFG-DOP parser only to predict the best tree for the input string. In other words, we do not output an f-structure but rather use the f-structure constraints on each fragment to rule out bad parse trees and better estimate the relative likelihoods of good parse trees. In this situation, retaining all possible constraints in each fragment f-structure is not only unnecessary but may be counter-productive. Therefore, we suggest that it would be more useful to learn which fragment constraints help to differentiate between good and bad

parse trees and, correspondingly, which constraints do not help with the task at hand. We could then simply delete those constraints deemed not useful.

It is important to note that we envisage deleting constraints from LFG-DOP fragments rather than the underlying LFG representations. Consider, for example, the representation for the string *Mary sees John* in (8.21)(*a*) below. If we extract LFG-DOP fragment (*b*), where the subject and object NP positions are open substitution sites, we can delete the constraints on the object NP – which correspond to the terminal *John* in (*a*) – as shown. However, when we extract LFG-DOP fragment (*c*) representing NP $\longrightarrow$ *John* we must retain these attribute-value pairs as we do not know which grammatical functions this fragment will fulfill in any unseen input sentences. Thus, learning which features to retain and which to delete must take place over the LFG-DOP fragment set rather than the LFG-DOP representations. Furthermore, we emphasise that feature selection should ideally take the form of a data-driven learning process so that the language-independent nature of the LFG-DOP model is not compromised.



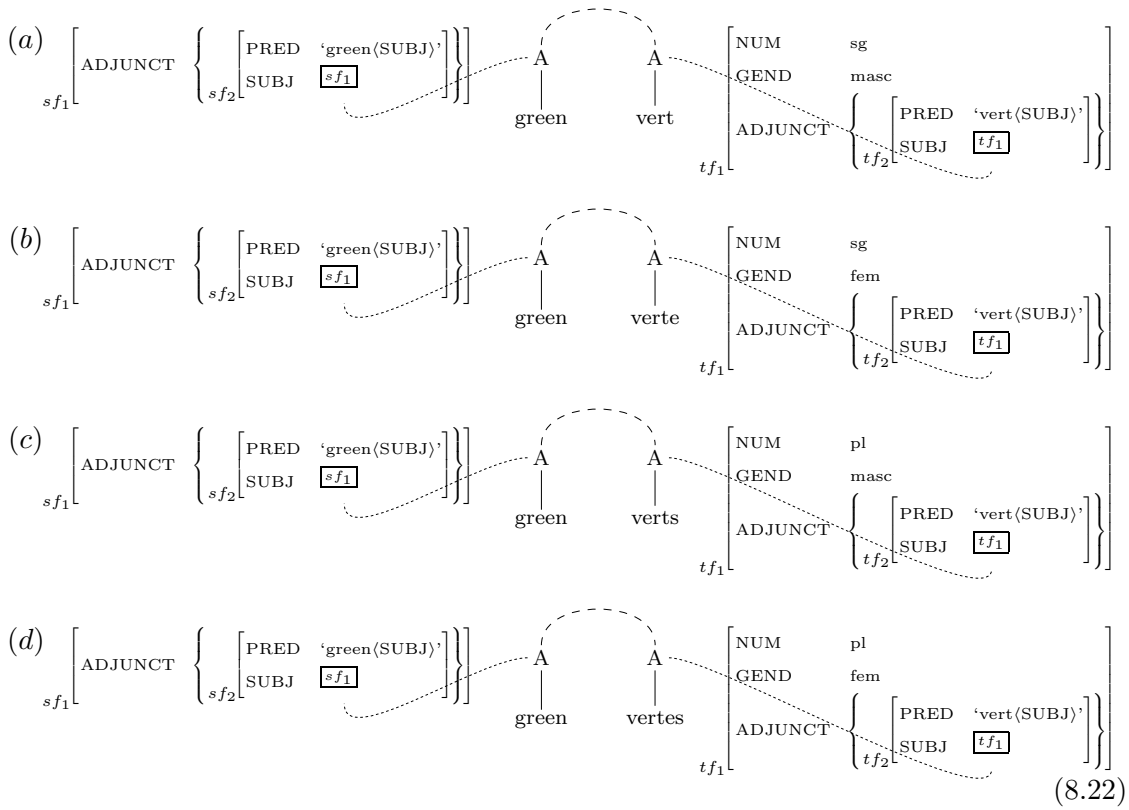
(8.21)

This type of LFG-DOP parser, which uses f-structure constraints to determine the best parse, is unlikely to be used for many real applications because LFG representations are generally considered to be more useful than representations which simply describe syntactic structure. This parsing model is, however, analogous to the LFG-DOT model

which uses f-structure constraints to determine the best translation. As the LFG-DOT representations for the source and target strings are internal to the model, we do not seek to build the most informative representations possible. Rather, we are interested in making use of only those constraints which will help to rule out bad translations and to better establish the relative likelihoods of good translations. Thus, we expect application of this strategy in the LFG-DOT model to help both translation speed and translation accuracy.

As the source sides of the LFG-DOT fragments extracted are a subset of those which would be extracted from the source side of the bilingual data according to the LFG-DOP model, and the same for the target sides, we suggest that the appropriate features could be learned on a monolingual basis. Consider, for example, fragments (a) – (d) in (8.22) below. We would hope to have learned that, for English, the surface form of the adjective *green* does not reflect whether the noun it modifies is singular or plural, i.e. both *the green leaves* and *the green leaf* are grammatically correct. Conversely, we would hope to have learned that, for French, the surface form of the adjective *vert* reflects both the number and gender features of the noun it modifies, meaning that this adjective has four possible surface forms. However, although no constraints are expressed in any of the f-structures corresponding to *green* other than that it must have a subject, the links between the English and French fragments in (8.22) express translational constraints. Thus, it is not the case that there are no constraints on the set of English adjective-noun pairs which can be analysed using each of these fragments. Rather, these constraints are translational rather than monolingual. For example, fragment (d) can only be used when the English word modified by *green*

translates as a French word with number plural and gender feminine.



(8.22)

While we have described feature pruning with respect to an LFG-DOP system which uses f-structure constraints to predict the best parse tree and to the LFG-DOT model which uses them to predict the best translation, it can also be applied where we wish to output the most informative LFG representation possible for the input string. In this situation, we determine those features which are helpful for disambiguation and mark them as *constraining* features. However, rather than deleting those features which do not appear to be helpful, we mark them as being *informative* rather than constraining. Thus, where a clash occurs during unification between values where one attribute-value pair has been identified as constraining and the other as informative, we can allow the constraining feature to take precedence and drop the feature which is merely informative. Furthermore, this representation can still be considered well-formed. This is in contrast to the current model, which offers no motivated way to decide which clashing feature should be deleted, and so deletes features indiscriminately and marks all resulting representations as ill-formed. Finally, in the situation where a clash occurs between two constraining equations and no representation can be generated without such clashes, discard can be applied as

before and the string flagged as ungrammatical with respect to the corpus.

8.6 Summary

In the first section of this chapter, we described the work which has been carried out to date on the LFG-DOT model of translation which augments Tree-DOT with LFG functional information. We then proposed an alternative LFG-DOT model which is based on Way (2001)'s LFG-DOT Model 3 but incorporates a different probability model and fragmentation procedure. We proposed that the implementation of this model follow the implementation of the LFG-DOP model, with the extension to handle paired source and target representations treated exactly as in Tree-DOT. We highlighted the differences between the contextual information captured by Tree-DOT and LFG-DOT representations and showed how these differences allow us to express different sets of translational dependencies between the same pairs of strings. Furthermore, we discussed how these differences impact on the issue of limited fragment compositionality for the Tree-DOT and LFG-DOT models. Finally, we discussed how both the LFG-DOP and LFG-DOT models could be improved by learning which attribute-value pairs contribute to the prediction of good solutions.

Chapter 9

Conclusions

In this thesis, we have provided a comprehensive description of the work carried out to date for tree-based Data-Oriented Parsing in terms of the model itself, pruning strategies, disambiguation techniques, parameter estimation methods and algorithms for improved efficiency. Furthermore, we have discussed in detail the algorithms used to implement each major component of our Tree-DOP parser. In addition, we have presented an empirical study of the characteristics of the Tree-DOP model when applied to parsing the English and French sections of the HomeCentre corpus. This investigation showed for this data that:

- DOP improves over the basic PCFG parsing model;
- highest parse accuracy is achieved by searching for the most probable parse rather than the most probable derivation or shortest derivation;
- it is no more time-consuming to find the most probable parse than to find the most probable or shortest derivations;
- overall, DOP modelled our English data better than it modelled our French data;
- the bias induced by the relative frequency parameter estimation method manifests itself as a contradiction of the DOP Hypothesis, i.e. as fragment depth increases, it becomes easier to determine which parse is most probable according to the model but the parses selected are of decreasing quality.

We have also presented the theoretical characteristics of the Data-Oriented Translation model, situating it as a hybrid model of MT which is unique in that it interweaves the philosophies of the rule-based, example-based and statistical approaches in an integrated framework. This model has the capacity to combine the linguistic sophistication of rule-based models of translation with the robustness and adaptability of data-driven methods and, thus, appears worthy of further research. However, previous attempts to evaluate the empirical characteristics of this model were hampered by unsuitable data and a limited implementation. Thus, we have presented the implementation details of our novel DOT system, which was inspired by the innovative algorithms developed for DOP, and have documented larger-scale, more translationally-complex experiments than heretofore. Contrary to previous findings, our results show that the model lives up to its theoretical promise. This empirical evaluation showed for the English-French HomeCentre corpus that:

- the DOT model performs significantly better than suggested by the previous evaluation;
- the DOT model outperforms the SMT system we trained and tested on the same data;
- highest translation accuracy is achieved by searching for the shortest derivation rather than the most probable translation, most probable parse or most probable derivation;
- the bias induced by the relative frequency parameter estimation method is less harmful to the DOT model than to DOP due to constrained fragmentation and pruning by link depth, but we still expect improved translation accuracy when satisfactory estimation techniques are applied to DOT;
- automated data-acquisition for DOT is a real possibility and is deserving of further attention.

The expressive power of the DOP model is limited by the corpus representations it assumes, and phrase-structure trees reflect surface syntactic phenomena only. We presented

the theoretical and practical work which has been carried out to date for the LFG-DOP model – which allies DOP with LFG representations – and summarised the empirical findings. We proposed an alternative definition for fragment extraction for this model, which addresses the handling of circular and re-entrant constraints and the (undesirable) retention of non-governable attributes for which there is no c-structure evidence. We discussed an alternative method for parameter estimation which extends the back-off re-estimation method described for Tree-DOP, highlighting not only how this approach addresses the model bias towards larger parse trees but also how it provides a motivated way to apply discard-generated fragments. Finally, we discussed how the LFG-DOP probability model limits the choice of implementation methodologies which can be employed.

Way (1999, 2001) investigated the possibility of merging the DOT model of translation with LFG representations. We have described the LFG-DOT models he proposed, and presented an alternative model which incorporates the novel definition we proposed for LFG-DOP fragment extraction and a different probability model. We suggested how this model might best be implemented, based on our findings with regard to the implementation of the DOT and LFG-DOP models. Furthermore, we explored the relationship between translational equivalence and limited generalisation reusability for both the tree-based and LFG-based DOT models, focussing on how this relationship differs depending on which formalism is assumed. Finally, we hypothesised as to how the constraints used to predict both good parses and good translations might be pruned in a motivated fashion.

9.1 Future work

Our empirical evaluation of the Tree-DOP model demonstrates the harmful effects of the bias induced by use of the relative frequency parameter estimation method. While this bias appears to be less harmful for the Tree-DOT model, we hypothesise that the application of parameter re-estimation using back-off will lead to further improvements in translation quality. Thus, we propose that experiments assessing the impact of applying this methodology to the Tree-DOT model be carried out in order to verify our hypothesis.

Our evaluation of Tree-DOT has shown that this model achieves high levels of translation accuracy. However, the problem of data acquisition constitutes a serious bottleneck as

the model requires that parsed sentence pairs be aligned at sentential and sub-structural levels. Manually establishing sub-structural alignments is impractical because it is time-consuming and requires considerable expertise of both source and target languages as well as how they are related. We have presented preliminary results which indicate that high-quality translations can also be achieved using automatically-induced alignments. Consequently, we feel that automatic acquisition of the resources required by the DOT model is a real possibility and deserves further attention.

While we have discussed LFG-DOP and LFG-DOT implementation possibilities, empirical evaluation of these models was beyond the scope of this thesis. While LFG-DOP experiments have already been documented in the literature, practical assessment of the LFG-DOT model of translation remains outstanding. We suggest that an empirical comparison of translation performance be carried out for the Tree-DOT and LFG-DOT models. We would also like to assess how the novel fragmentation methodology we have proposed impacts on the accuracy and efficiency of the LFG-based models of both parsing translation, particularly with regard to the handling of sentences involving circular and re-entrant structures. We would like to evaluate the back-off parameter estimation technique proposed for LFG-DOP, paying particular attention to how structuring the space of discard fragments helps to find accurate solutions more quickly. We hypothesise that this method of estimating fragment probabilities is also applicable to LFG-DOT. Finally, we propose that a learning method to distinguish between constraining and informative fragment features for both LFG-DOP and LFG-DOT be developed, and experiments carried out to establish the impact this has on accuracy and efficiency.

Bibliography

Abeillé, A., Schabes, Y., and Joshi, A. K. (1990). Using Lexicalized Tags for Machine Translation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90)*, pages 1–6, Helsinki, Finland.

Abney, S. (1997). Stochastic Attribute-Value Grammars. *Computational Linguistics*, **23**(4):597–618.

Aho, A. and Ullman, J. (1972). *The Theory of Parsing, Translation and Compiling. Volume 1: Parsing*. Prentice-Hall, Englewood Cliffs, NJ.

Bod, R. (1992). A Computational Model of Language Performance: Data Oriented Parsing. In *Proceedings of the 15th[sic] International Conference on Computational Linguistics (COLING'92)*, pages 855–859, Nantes, France.

Bod, R. (1995a). *Enriching Linguistics with Statistics: Performance Models of Natural Language*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam.

Bod, R. (1995b). The Problem of Computing the Most Probable Tree in Data-Oriented Parsing and Stochastic Tree Grammars. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95)*, pages 104–111, Dublin, Ireland.

Bod, R. (1996). Monte Carlo Parsing. In Bunt, H. and Tomita, M., editors, *Recent Advances in Parsing Technology*, pages 255–280. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Bod, R. (1998). *Beyond Grammar: An Experience-Based Theory of Language.* Stanford CA: CSLI Publications.

Bod, R. (1999). Extracting Stochastic Grammars from Treebanks. In *Proceedings of the ATALA Workshop on Treebanks*, Paris, France.

Bod, R. (2000a). An Empirical Evaluation of LFG-DOP. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 62–68, Saarbrucken, Germany.

Bod, R. (2000b). An Improved Parser for Data-Oriented Lexical-Functional Analysis. In *Proceedings of the 38th Conference of the Association for Computational Linguistics*, pages 61–68, Hong Kong.

Bod, R. (2000c). Combining Semantic and Syntactic Structure for Language Modeling. In *Proceedings of the Sixth International Conference on Spoken Language Processing (ICSLP 2000)*, volume **3**, pages 106–109, Beijing, China.

Bod, R. (2000d). Context-Sensitive Spoken Dialogue Processing with the DOP Model. *Natural Language Engineering*, **5(4)**:309–323.

Bod, R. (2000e). Parsing with the Shortest Derivation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING'00)*, pages 69–75, Saarbruecken, Germany.

Bod, R. (2001). What is the Minimal Set of Fragments that Achieves Maximal Parse Accuracy? In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01)*, pages 66–73, Toulouse, France.

Bod, R. (2002). A Unified Model of Structural Organisation in Language and Music. *Journal of Artificial Intelligence Research*, **17**:289–308.

Bod, R. (2003a). An Efficient Implementation of a New DOP Model. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, pages 19–26, Budapest, Hungary.

Bod, R. (2003b). Do All Fragments Count? *Natural Language Engineering*, **9(4)**:307–323.

Bod, R. (2003c). Extracting Stochastic Grammars from Treebanks. In Abeillé, A., editor, *Treebanks: Building and Using Parallel Corpora*, pages 333–350. Kluwer Academic Publishers.

Bod, R. and Kaplan, R. (1998). A Probabilistic Corpus-Driven Model for Lexical-Functional Analysis. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Conference of the Association for Computational Linguistics*, pages 145–151, Montreal, Canada.

Bod, R. and Kaplan, R. (1999). A Probabilistic Corpus-Driven Approach to Lexical-Functional Representations. Unpublished manuscript.

Bod, R. and Kaplan, R. (2003). A DOP model for Lexical-Functional Grammar. In Bod, R., Scha, R., and Sima'an, K., editors, *Data-Oriented Parsing*, pages 211–232. Stanford CA: CSLI Publications.

Bod, R., Scha, R., and Sima'an, K., editors (2003). *Data-Oriented Parsing*. Stanford CA: CSLI Publications.

Bond, F. and Shirai, S. (2003). A Hybrid Rule and Example-Based Method for Machine Translation. In Carl, M. and Way, A., editors, *Recent Advances in Example-Based Machine Translation*, pages 211–224. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Bonnema, R., Buying, P., and Scha, R. (2000). Parse Tree Probability in Data Oriented Parsing. In *Proceedings of the Conference on Intelligent Text Processing and Computational Linguistics*, pages 219–232, Mexico City, Mexico.

Bonnema, R. and Scha, R. (2003). Reconsidering the Probability Model for DOP. In Bod, R., Scha, R., and Sima'an, K., editors, *Data-Oriented Parsing*, pages 25–42. Stanford CA: CSLI Publications.

Bresnan, J. (2001). *Lexical-Functional Syntax*. Blackwell, Oxford.

Brown, R. D. (2003). Clustered Transfer-Rule Induction for Example-Based Translation.

In Carl, M. and Way, A., editors, *Recent Advances in Example-Based Machine Translation*, pages 59–82. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Carroll, G. and Rooth, M. (1998). Valence induction with a head-lexicalized PCFG. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing (EMNLP 3)*, pages 36–45, Granada, Spain.

Chappelier, J.-C. and Rajman, M. (2003). Parsing DOP with Monte-Carlo Techniques. In Bod, R., Scha, R., and Sima'an, K., editors, *Data-Oriented Parsing*, pages 83–106. Stanford CA: CSLI Publications.

Charniak, E., Knight, K., and Yamada, K. (2003). Syntax-based Language Models for Machine Translation. In *Proceedings of the Ninth Machine Translation Summit*, pages 40–46, New Orleans, USA.

Clarkson, P. and Rosenfeld, R. (1997). Statistical Language Modeling Using the CMU–Cambridge Toolkit. In *Proceedings of the 5th biennial European Conference on Speech Communication and Technology (EUROSPEECH'97)*, pages 2707–2710, Rhodes, Greece.

Collins, M. and Duffy, N. (2001). Convolution Kernels for Natural Language. In *Advances in Neural Information Processing Systems 14*. MIT Press.

Cormons, B. (1999). *Analyse et Disambiguation: Une approche purement à base de corpus (Data-Oriented Parsing) pour le formalisme des Grammaires Lexicales Fonctionelles*. PhD thesis, Université de Rennes, France.

Daelemans, W. (1999). Memory-Based Language Processing. *Journal for Experimental and Theoretical Artificial Intelligence*, **11**(3):287–467.

Dalrymple, M. (2001). *Lexical-Functional Grammar*. Academic Press, San Diego, CA; London.

de Pauw, G. (2003). An Approximation of DOP through Memory-Based Learning. In Bod, R., Scha, R., and Sima'an, K., editors, *Data-Oriented Parsing*, pages 147–167. Stanford CA: CSLI Publications.

Doddington, G. (2002). Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. In *Human Language Technology: Notebook Proceedings*, pages 128–132, San Diego, CA.

Earley, J. (1970). An Efficient Context-Free Parsing Algorithm. *Communications of the ACM*, **6**(8):451–455.

Finch, A., Watanabe, T., and Sumita, E. (2003). Data-Oriented Paraphrasing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP'03)*, pages 153–157, Borovets, Bulgaria.

Frank, A. (1999). LFG-based syntactic transfer from English to French with the Xerox Translation Environment. In *Proceedings of the ESSLLI'99 Summer School*, Utrecht, The Netherlands.

Germann, U. (2003). Greedy Decoding for Statistical Machine Translation in Almost Linear Time. In *Proceedings of the Joint Meeting of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 72–79, Edmonton, Canada.

Germann, U., Jahr, M., Knight, K., Marcu, D., and Yamada, K. (2001). Fast Decoding and Optimal Decoding for Machine Translation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01)*, pages 228–235, Toulouse, France.

Goodman, J. (1996a). Efficient Algorithms for Parsing the DOP model. In *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing (EMNLP 1)*, pages 143–152, Philadelphia, PA.

Goodman, J. (1996b). Parsing Algorithms and Metrics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pages 177–183, Santa Cruz, CA.

Goodman, J. (1998). *Parsing inside-out*. PhD thesis, Harvard University, MA.

Goodman, J. (2003). Efficient Parsing of DOP with PCFG-Reductions. In Bod, R., Scha, R., and Sima'an, K., editors, *Data-Oriented Parsing*, pages 125–146. Stanford CA: CSLI Publications.

Groves, D., Hearne, M., and Way, A. (2004). Robust Sub-Sentential Alignment of Phrase-Structure Trees. In *Proceedings of The 20th International Conference on Computational Linguistics (COLING'04)*, pages 1072–1078, Geneva, Switzerland.

Hearne, M. and Sima'an, K. (2003). Structured Parameter Estimation for LFG-DOP using Backoff. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP'03)*, pages 184–191, Borovets, Bulgaria.

Hearne, M. and Way, A. (2003). Seeing the Wood for the Trees: Data-Oriented Translation. In *Proceedings of the Ninth Machine Translation Summit*, pages 165–172, New Orleans, USA.

Hearne, M. and Way, A. (2004). Data-Oriented Parsing and the Penn Chinese Treebank. In *Proceedings of the 1st International Joint Conference on Natural Language Processing (IJCNLP'04)*, pages 406–413, Hainan Island, China.

Hoogweg, L. (2000). Extending DOP1 with the insertion operation. Master's thesis, University of Amsterdam, The Netherlands.

Hutchins, J. and Somers, H. (1992). *An Introduction to Machine Translation*. Academic Press, London.

Johnson, M. (1999). PCFG models of linguistic tree representations. *Computational Linguistics*, **24**(4):613–632.

Johnson, M. (2002). The DOP estimation method is biased and inconsistent. *Computational Linguistics*, **28**(1):71–76.

Kaplan, R. and Bresnan, J. (1982). Lexical Functional Grammar, a Formal System for Grammatical Representation. In Bresnan, J., editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 110–115, Princeton, NJ.

Melamed, I. D. (2004). Statistical Machine Translation by Parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, pages 653–660, Barcelona, Spain.

Melamed, I. D., Green, R., and Turian, J. P. (2003). Precision and Recall of Machine Translation. Technical Report 03-004, New York University, NY.

Menezes, A. and Richardson, S. D. (2003). A Best-First Alignment Algorithm for Extraction of Transfer Mappings. In Carl, M. and Way, A., editors, *Recent Advances in Example-Based Machine Translation*, pages 421–442. Kluwer Academic Publishers, Dordrecht, The Netherlands.

NIST (2002). Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. Technical report.

Och, F. J. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2001). BLEU: a Method for Automatic Evaluation of Machine Translation. Technical report, IBM T.J. Watson Research Center.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 311–318, Philadelphia, PA.

Poutsma, A. (1998). Data-Oriented Translation. In *Ninth Conference of Computational Linguistics in the Netherlands*, Leuven, Belgium.

Poutsma, A. (2000). Data-Oriented Translation: Using the Data-Oriented Parsing framework for Machine Translation. Master's thesis, University of Amsterdam, The Netherlands.

Poutsma, A. (2003). Machine Translation with Tree-DOP. In Bod, R., Scha, R., and Sima'an, K., editors, *Data-Oriented Parsing*, pages 339–357. Stanford CA: CSLI Publications.

Sato, S. (1995). MBT2: a method for combining fragments of examples in example-based translation. *Artificial Intelligence*, **75**:31–49.

Scha, R. (1990). Language Theory and Language Technology: Competence and Performance. *Computertoepassingen in de Neerlandistiek*, pages 7–22.

Sima'an, K. (1995a). An optimized algorithm for Data Oriented Parsing. In *Proceedings of International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, Bulgaria.

Sima'an, K. (1995b). Computational Complexity of Probabilistic Disambiguation by means of Tree-Grammars. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'96)*, pages 1175–1180, Copenhagen, Denmark.

Sima'an, K. (1999). *Learning Efficient Disambiguation*. PhD thesis, University of Amsterdam, The Netherlands.

Sima'an, K. (2003). Computational Complexity of Disambiguation under DOP1. In Bod, R., Scha, R., and Sima'an, K., editors, *Data-Oriented Parsing*, pages 63–81. Stanford CA: CSLI Publications.

Sima'an, K. and Buratto, L. (2003). Backoff Parameter Estimation for the DOP Model. In *Proceedings of the 14th European Conference on Machine Learning (ECML'03)*, pages 373–384, Cavtat-Dubrovnik, Croatia.

Stolcke, A. (1995). An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities. *Computational Linguistics*, **21**(2):165–201.

Turcato, D. and Popowich, F. (2003). What is Example-Based Machine Translation? In Carl, M. and Way, A., editors, *Recent Advances in Example-Based Machine Translation*, pages 59–82. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Turian, J. P., Shen, L., and Melamed, I. D. (2003). Evaluation of Machine Translation and its Evaluation. Technical Report 03-005, New York University, NY.

Vauquois, B. (1968). A Survey of Formal Grammars and Algorithms for Recognition and Transformation in Machine Translation. In *IFIP Congress-68*, pages 254–260, Edinburgh.

Way, A. (1999). A Hybrid Architecture for Robust MT using LFG-DOP. *Journal of Experimental and Theoretical Artificial Intelligence*, **11**:441–471.

Way, A. (2001). *LFG-DOT: A Hybrid Architecture for Robust MT*. PhD thesis, University of Essex, Colchester, UK.

Way, A. and Gough, N. (2003). wEBMT: Developing and Validating an Example-Based Machine Translation System using the World Wide Web. *Computational Linguistics: Special Issue on the Web as Corpus*, **29**(3):421–458.

Wu, D. (1997). Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–404.

Yamada, K. and Knight, K. (2001). A Syntax-Based Statistical Translation Model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01)*, pages 523–530, Toulouse, France.

Younger, D. (1967). Recognition and parsing of context-free languages in time n^3. *Information Control*, **10**(2):189–208.