THE UNSUPERVISED LEARNING OF NATURAL LANGUAGE STRUCTURE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Dan Klein
March 2005

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christopher D. Manning
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daniel Jurafsky

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daphne Koller

Approved for the University Committee on Graduate Studies.

To my mom, who made me who I am
and Jennifer who liked me that way.

Acknowledgments

Many thanks are due here. First, to Chris Manning for being a fantastic, insightful advisor. He shaped how I think, write, and do research at a very deep level. I learned from him how to be a linguist and a computer scientist at the same time (he's constantly in both worlds) and how to see where an approach will break before I even try it. His taste in research was also a terrific match to mine; we found so many of the same problems and solutions compelling (even though our taste in restaurants can probably never be reconciled). Something I hope to emulate in the future is how he always made me feel like I had the freedom to work on whatever I wanted, but also managed to ensure that my projects were always going somewhere useful. Second, thanks to Daphne Koller, who gave me lots of good advice and support over the years, often in extremely dense bursts that I had to meditate on to fully appreciate. I'm also constantly amazed at how much I learned in her cs228 course. It was actually the only course I took at Stanford, but after taking that one, what else do you need? Rounding out my reading committee, Dan Jurafsky got me in the habit of asking myself, "What is the rhetorical point of this slide?" which I intend to ask myself mercilessly every time I prepare a presentation. The whole committee deserves extra collective thanks for bearing with my strange thesis timeline, which was most definitely not optimally convenient for them. Thanks also to the rest of my defense committee, Herb Clark and David Beaver.

Many other people's research influenced this work. High on the list is Alex Clark, whose distributional syntax paper in CoNLL 2001 was so similar in spirit to mine that when my talk was scheduled immediately after his, I had the urge to just say, "What he said." Lots of others, too: Menno van Zaanen, Deniz Yuret, Carl de Marcken, Shimon Edelman, and Mark Paskin, to name a few. Noah Smith deserves special mention both for

pointing out to me some facts about tree distributions (see the appendix) and for independently reimplementing the constituent-context model, putting to rest the worry that some beneficial bug was somehow running the show. Thanks also to Eugene Charniak, Jason Eisner, Joshua Goodman, Bob Moore, Fernando Pereira, Yoram Singer, and many others for their feedback, support, and advice on this and related work. And I'll miss the entire Stanford NLP group, which was so much fun to work with, including officemates Roger Levy and Kristina Toutanova, and honorary officemates Teg Grenager and Ben Taskar.

Finally, my deepest thanks for the love and support of my family. To my grandfathers Joseph Klein and Herbert Miller: I love and miss you both. To my mom Jan and to Jenn, go read the dedication!

Abstract

There is precisely one complete language processing system to date: the human brain. Though there is debate on how much built-in bias human learners might have, we definitely acquire language in a primarily unsupervised fashion. On the other hand, computational approaches to language processing are almost exclusively supervised, relying on hand-labeled corpora for training. This reliance is largely due to unsupervised approaches having repeatedly exhibited discouraging performance. In particular, the problem of learning syntax (grammar) from completely unannotated text has received a great deal of attention for well over a decade, with little in the way of positive results. We argue that previous methods for this task have generally underperformed because of the representations they used. Overly complex models are easily distracted by non-syntactic correlations (such as topical associations), while overly simple models aren't rich enough to capture important first-order properties of language (such as directionality, adjacency, and valence).

In this work, we describe several syntactic representations and associated probabilistic models which are designed to capture the basic character of natural language syntax as directly as possible. First, we examine a nested, distributional method which induces bracketed tree structures. Second, we examine a dependency model which induces word-to-word dependency structures. Finally, we demonstrate that these two models perform better in combination than they do alone. With these representations, high-quality analyses can be learned from surprisingly little text, with no labeled examples, in several languages (we show experiments with English, German, and Chinese). Our results show above-baseline performance in unsupervised parsing in each of these languages.

Grammar induction methods are useful since parsed corpora exist for only a small number of languages. More generally, most high-level NLP tasks, such as machine translation

and question-answering, lack richly annotated corpora, making unsupervised methods extremely appealing even for common languages like English. Finally, while the models in this work are not intended to be cognitively plausible, their effectiveness can inform the investigation of what biases are or are not needed in the human acquisition of language.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Problem of Learning a Language

The problem of how a learner, be it human or machine, might go about acquiring a human language has received a great deal of attention over the years. This inquiry raises many questions, some regarding the human language acquisition process, some regarding statistical machine learning approaches, and some shared, relating more to the structure of the language being learned than the learner. While this chapter touches on a variety of these questions, the bulk of this thesis focuses on the unsupervised machine learning of a language's syntax from a corpus of observed sentences.

1.1.1 Machine Learning of Tree Structured Linguistic Syntax

This work investigates learners which induce hierarchical syntactic structures from observed yields alone, sometimes referred to as *tree induction*. For example, a learner might observe the following corpus:

the cat stalked the mouse
the mouse quivered
the cat smiled

Given this data, the learner might conclude that *the mouse* is some kind of unit, since it occurs frequently and in multiple contexts. Moreover, the learner might posit that *the cat* is

somehow similar to *the mouse*, since they are observed in similar contexts. This example is extremely vague, and its input corpus is trivial. In later chapters, we will present concrete systems which operate over substantial corpora.

Compared to the task facing a human child, this isolated syntax learning task is easier in some ways but harder in others. On one hand, natural language is an extremely complex phenomenon, and isolating the learning of syntax is a simplification. A complete knowledge of language includes far more than the ability to group words into nested units. There are other components to syntax, such as sub-word morphology, agreement, dislocation/non-locality effects, binding and quantification, exceptional constructions, and many more. Moreover, there are crucial components to language beyond syntax, particularly semantics and discourse structure, but also (ordinarily) phonology. A tree induction system is not forced to simultaneously learn all aspects of language. On the other hand, the systems we investigate have far fewer cues to leverage than a child would. A child faced with the utterances above would generally know something about cats, mice, and their interactions, while, to the syntax-only learner, words are opaque symbols.

Despite being dissimilar to the human language acquisition process, the tree induction task has received a great deal of attention in the natural language processing and computational linguistics community (Carroll and Charniak 1992, Pereira and Schabes 1992, Brill 1993, Stolcke and Omohundro 1994). Researchers have justified isolating it in several ways. First, for researchers interested in arguing empirically against the poverty of the stimulus, whatever syntactic structure can be learned in isolation gives a bound on how much structure can be learned by a more comprehensive learner (Clark 2001a). Moreover, to the extent that the syntactic component of natural language is truly modular (Fodor 1983, Jackendoff 1996), one might expect it to be learnable in isolation (even if a human learner would never have a reason to). More practically, the processing task of parsing sentences into trees is usually approached as a stand-alone task by NLP researchers. To the extent that one cares about this kind of syntactic parsing as a delimited task, it is useful to learn such structure as a delimited task. In addition, learning syntax without either presupposing or jointly learning semantics may actually make the task easier, if less organic. There is less to learn, which can lead to simpler, more tractable machine learning models (later chapters argue that this is a virtue).

1.1.2 Inducing Treebanks and Parsers

There are practical reasons to build tree induction systems for their own sakes. In particular, one might reasonably be interested in the learned artifact itself – a parser or grammar. Nearly all natural language parsing is done using supervised learning methods, whereby a large treebank of hand-parsed sentences is generalized to new sentences using statistical techniques (Charniak 1996). This approach has resulted in highly accurate parsers for English newswire (Collins 1999, Charniak 2000) which are trained on the Penn (English) Treebank (Marcus et al. 1993). Parsers trained on English newswire degrade substantially when applied to new genres and domains, and fail entirely when applied to new languages. Similar treebanks now exist for several other languages, but each treebank requires many person-years of work to construct, and most languages are without such a resource. Since there are many languages, and many genres and domains within each language, unsupervised parsing methods would represent a solution to a very real resource constraint.

If unsupervised parsers equaled supervised ones in accuracy, they would inherit all the applications supervised parsers have. Even if unsupervised parsers exhibited more modest performance, there are plenty of ways in which their noisier output could be useful. Induction systems might be used as a first pass in annotating large treebanks (van Zaanen 2000), or features extracted from unsupervised parsers could be a "better than nothing" stop-gap for systems such as named-entity detectors which can incorporate parse features, but do not require them to be perfect. Such systems will simply make less use of them if they are less reliable.

1.1.3 Learnability and the Logical Problem of Language Acquisition

Linguists, philosophers, and psychologists have all considered the *logical problem of language acquisition* (also referred to as *Plato's problem*) (Chomsky 1965, Baker and McCarthy 1981, Chomsky 1986, Pinker 1994, Pullum 1996). The logical (distinct from empirical) problem of language acquisition is that a child hears a finite number of utterances from a target language. This finite experience is consistent with infinitely many possible targets. Nonetheless, the child somehow manages to single out the correct target language. Of course, it is not true that every child learns their language perfectly, but the key issue is

that they eventually settle on the correct generalizations of the evidence they hear, rather than wildly incorrect generalizations which are equally consistent with that evidence.

A version of this problem was formalized in Gold (1967). In his formulation, we are given a target language L drawn from a set $\mathcal{L}$ of possible languages. A learner C is shown a sequence of positive examples $[s_i], s_i \in L$ – that is, it is shown grammatical utterances.[1] However, the learner is never given negative examples, i.e., told that some s is ungrammatical ($s \notin L$). There is a guarantee about the order of presentation: each $s \in L$ will be presented at some point i. There are no other guarantees on the order or frequency of examples.

The learner C maintains a hypothesis $L(C, [s_0 \ldots s_i]) \in \mathcal{L}$ at all times. Gold's criterion of learning is the extremely strict notion of *identifiability in the limit*. A language family $\mathcal{L}$ is identifiable in the limit if there is some learner C such that, for any $L \in \mathcal{L}$ and any legal presentation of examples $[s_i]$, there is some point k such that for all $j > k$, $L(C, [s_0 \ldots s_k]) = L$. In other words, for any target language and example sequence, the learner's hypothesis is eventually correct (whether the learner knows it or not). For example, the family $\mathcal{L} = \{\{a\}, \{a, b\}\}$ is learnable by the following algorithm: initially posit $\{a\}$, and switch to $\{a, b\}$ upon being presented with a b example. The learner is either correct from the start, or correct as soon as a b example occurs (which is guaranteed).

Gold's famous results show that a wide variety of language families are not learnable in this strict sense. In particular, any *superfinite* family, i.e., a family which contains all the finite languages and at least one infinite language, is not learnable. Since the family of regular languages is superfinite, regular languages aren't identifiable in the limit. Therefore, neither are context-free languages. This result has often been taken as a strong argument against practical learnability of human language.

As stated here, Gold's formalization is open to a wide array of basic objections. First, as mentioned above, who knows whether all children in a linguistic community actually do learn exactly the same language? All we really know is that their languages are similar enough to enable normal communication. Second, for families of probabilistic languages,

[1] "Grammatical" is a loaded term, but is intended to capture the partially pre-theoretical distinction between utterances the learner should accept as well-formed at the end of a successful learning process.

why not assume that the examples are sampled according to the target language's distribution? Then, while a very large corpus won't contain every sentence in the language, it can be expected to contain the common ones. Indeed, while the family of context-free grammars is unlearnable in the Gold sense, Horning (1969) shows that a slightly softer form of identification is possible for the family of probabilistic context-free grammars if these two constraints are relaxed (and a strong assumption about priors over grammars is made).

Another objection one can raise with the Gold setup is the absence of negative examples. Negative feedback might practically be very powerful, though formal results such as Angluin (1990) suggest that allowing negative feedback doesn't completely solve the problem. They consider the addition of an *equivalence oracle*, which allows the learner to present a hypothesis and get a counterexample if that hypothesis is incorrect. Even with such an oracle, the class of context-free grammars is not identifiable in polynomial time. The issue of negative feedback is often raised in conjunction with child language acquisition, where a perennial debate rages as to whether children receive negative feedback, and what use they make of it if they do (Brown and Hanlon 1970, Marcus 1993). A strong form of negative feedback would be explicit correction – where the child utters examples from their hypothesized language L' and a parent maps those examples into related examples from the correct language L. There is a large body of evidence that children either do not receive explicit correction or do not make good use of it when they do (Hirsh-Pasek et al. 1984, Demetras et al. 1986, Penner 1986). A weaker form of negative feedback is where the child utters examples from L', and, if the example is not a well-formed element of L (with the same meaning), the attempted communication is unsuccessful. This kind of feedback seems plausible, and even bears a resemblance to Angluin's equivalence queries. It also has the advantage that the notion of "related" that maps ungrammatical queries to grammatical ones, which would be a highly semantic and contextual process, need not be specified.

1.1.4 Nativism and the Poverty of the Stimulus

An issue that linguists, and others, have spent a great deal of energy arguing for (and against) is Chomsky's hypothesis of the *poverty of the stimulus* (Chomsky 1965). The logical problem of language acquisition is, basically, the problem that children make judgments about examples they haven't seen, based on examples that they have. This necessitates a process of generalization. Chomsky's argument goes along these lines: children learn subtle facts about their language from data which (putatively) does not contain evidence for or against those facts. The problem of the poverty of the stimulus refers to the lack of crucial relevant data in the learner's experience. Chomsky's solution is to appeal to a richness of constraint. He argues that because human languages are highly constrained, the actual family of human languages is relatively small (perhaps because of the bias in evolved special-purpose hardware). Therefore small amounts of data suffice for a learner to single out a target language. Down this road often lies strong nativist argumentation, but the source of such constraints is really an orthogonal issue.

Chomsky also takes a strong position arguing that human language is a symbolic phenomenon, as opposed to a probabilistic one (Chomsky 1965, Chomsky 1986). That is, there are, of course, trends where we actually say one thing more or less often than some other thing, but these facts are epiphenomenal to a human's knowledge of a language. This viewpoint is fairly far removed from the viewpoint of this thesis, in which (excepting chapter 4) the knowledge of syntax is encoded in the parameters of various probabilistic models. The successes of these kinds of systems in recovering substantial portions of the broad structure of a language do indicate that the probabilistic trends can be pronounced, detectable, and usefully exploited. However, such results only serve as indirect evidence for or against nativism and symbolic theories of language.

1.1.5 Strong vs. Weak Generative Capacity

A useful contrast in linguistic theory is the distinction between the *weak* and *strong generative capacity* of a grammar (Miller 1999). The weak generative capacity of a grammar is the set of utterances it allows. The strong generative capacity, on the other hand, is the set of derivations it allows. Two grammars may have the same weak capacity – generate

the same set of utterances – but have different strong capacities. For example, consider the following two grammars:

S → NP VP	S → VP NP
VP → V NP	VP → NP VP
(a)	(b)

From their start symbols S, both grammars produce (only) the subject-verb-object sequence NP V NP, and therefore have the same weak generative capacity. However, grammar (a) does so using a traditional verb-object VP structure, while grammar (b) uses a subject-verb group, so their strong capacities are different. To the extent that we just want to predict that NP V NP is a valid English sequence while NP NP V is not, either grammar suffices. If we care about the tree structures, we may well prefer one grammar over the other; in this case, a variety of linguistic evidence has led to the general preference of the left grammar over the right one.

In a probabilistic context, the weak capacity (in the strict symbolic sense) of a grammar is often uninteresting, since many probabilistic models accept all terminal sequences with some (possibly very low) probability. Models within the same representational family will also often accept all derivations, again with possibly vanishing probabilities. In this case, the straightforward softenings of the weak and strong capacities of a probabilistic model are the densities that the model assigns to specific derivations (strong capacity), and utterances (weak capacity).

One can have varying degrees of interest in the strong vs. weak capacities of a probabilistic model. The weak capacity – density over utterances – is the primary prediction of interest in language modeling tasks, such as for noisy-channel speech or translation models (Chelba and Jelinek 1998, Charniak et al. 2003). Some work on grammar induction has specifically aimed to learn good language models in this sense, for example (Baker 1979, Chen 1995). Note that, to the extent that one is interested only in the weak capacity of a grammar, there is no need to built tree-structured models, or even to have any induced hidden structure at all. One can simply build fully-observed models, such as n-gram models. In this context, hidden structure, such as parse trees or part-of-speech chains, is only useful insofar as it enables a better model over the observed structure. In particular,

it is not necessary or important that the hidden structure induced correspond to linguistic preconceptions.

In contrast, if one is primarily interested in the induced structures themselves, such as if one is inducing a tree model with the intention of using induced trees to represent a certain kind of syntactic structure for use in subsequent processing, then the strong capacity becomes of primary interest. A minimal goal in this case is that the hidden structures postulated be consistent – for example, that learned trees either group the subject and verb together, or group the verb and object together, so long as the chosen analysis is consistent from sentence to sentence. A more ambitious goal is to aim for the recovery of linguistically plausible analyses, in which case we have the added preference for the traditional verb-object grouping. Of course, it is often far from clear which of several competing analyses is the linguistically correct one, but in many cases, such as with the verb-object grouping, particular analyses are supported by the convergence of a good variety of evidence.

In this work, we are interested in inducing grammar models for their strong capacity. The quality of induced structures will thus be evaluated by a mix of comparing how closely they replicate linguist-annotated treebanks (on the assumption that such treebanks are broadly correct) and error analysis of the discrepancies (both to illustrate true errors and to show acceptable behavior that deviates from the goal treebank).

It is important to note that there is at least one other goal one can have for a language learning system: the cognitively plausible modeling of human language acquisition. This is essentially an axis orthogonal to the strong/weak issue. In particular, if one wants to mimic a human learner in a weak way, one can try to mimic the utterances produced, for example, hoping that the ability to produce various constructions is manifested in the same order as for a human learner. On the other hand, one can try to reproduce the tree structures used by human learners, as well, though this requires a greater commitment to the reality of tree-structure syntax than some psychologists would like. Solan et al. (2003) is an example of a system which produces non-tree structured grammars, where the goal is cognitive plausibility, the structures themselves are of interest, but there is no desire to replicate traditional linguistic analyses. Such authors would likely criticize the present work as having the wrong objective: too much concern with recovering traditional linguistic structure, too little concern with human psychology.

To be clear on this point: the goal of this work is not to produce a psychologically plausible model or simulation. However, while success at the tree induction task does not directly speak to the investigation of the human language faculty, it does have direct relevance to the logical problem of language acquisition, particularly the argument of the poverty of the stimulus, and therefore an indirect relevance to cognitive investigations. In particular, while no such machine system can tell us how humans do learn language, it can demonstrate the presence and strength of statistical patterns which are potentially available to a human learner.

1.2 Limitations of this Work

This work has several limitations, some of which are purposeful and serve to usefully delimit the scope of the investigation and some of which are more problematic.

1.2.1 Assumptions about Word Classes

An intentional delimitation of the problem addressed is that the models in this work all assume that in addition to, or, more often instead of, a sequence of words, one has a sequence of word classes, for example a sequence of part-of-speech tags. There are several reasons for this assumption. First, and weakest, it is a traditional simplification, and a good deal of prior work begins at the word class level, usually because it counteracts sparsity and reduces the computational scale of most potential solutions. Second, prior work on part-of-speech induction (see section 3.3) has been successful enough that, even though jointly learning parts-of-speech and syntax is appealing, an appeal to previous work to provide initial word classes seems reasonable. Third, as we will argue in chapter 6, models over word classes are actually more likely to detect valid syntactic configurations in some cases, because a strong correlation between two specific words is more likely to be evidence of a topical relationship than a syntactic one. It is entirely possible that there is some advantage to inducing parts-of-speech jointly with higher level syntax, but for the present work we keep them separate as a hopefully defensible choice of scope and convenience.

1.2.2 No Model of Semantics

The most blatant difference between the task facing a child language learner and the systems presented here is that, for the child, language is highly situated. The utterances have meaning and communicative purpose, and all agents in the conversation have models of what the other agents are trying to accomplish. Utterances can be supplemented with other mechanisms of communication, such as deixis. Combinations of known words are constrained by the combinations of their semantics. There are other substantial differences, such as the gradual increase in complexity of communication over time for children, but the presence of meaning and intent is the most severe difference between human language acquisition and form-only machine learning from text.

Learning the syntax of utterances when the meaning of those utterances is already known is clearly an easier problem than learning syntax without such knowledge. This constrained learning has been explored in Chang and Gurevich (2004), for example. What is less clear is whether learning syntax at the same time as learning semantics is easier or harder than learning the syntax alone, the trade-off being between having a more complex model (which would tend to make induction more difficult) and having the ability to exploit orthogonal cues (which could make it easier).

In this work, we try to learn syntax alone, using observed utterances alone. This conception of the language learning task certainly has a long history, and can be defended on several grounds. First, results on this task inform the debate on the logical problem of language learning and innateness. A successful grammar induction system provides an important lower bound on the amount of bias required to recover the syntax of a language. Without serious cognitive modeling, it is difficult to argue that humans actually use the same kinds of statistical cues that these systems use to extract grammar from data (though see Saffran et al. (1996) for some evidence that statistical cues are used in word segmentation). However, it does show the degree to which those cues exist and it does argue that the human mechanism does not necessarily need to be more highly biased than the machine learner. In fact, to the degree that the machine learner is solving in isolation a problem that humans solve in a situated fashion, we would expect the machine learner to require greater bias that the human learner.

Second, and related, it is worth investigating how far one can go in learning syntax on its own. Empirical evidence suggests that some superficial components of language can be learned by human learners from purely structural evidence. For example, Saffran et al. (1996) shows that babies are capable of accurately segmenting streams of nonsense words on the basis of their statistical distributions, just from hearing the streams playing in the background for a short time. Of course, it could be that this ability to segment words distributionally is only a small component of the human word-segmentation faculty, and that it is even less of a component in the learning of deeper syntactic structures. However, it is still worth investigating how strong the cues are, regardless of whether such meaning-free learning is a partial or exclusive mechanism for human learners.

Third, the task of annotating raw text with syntactic structures is an important practical engineering task. The natural language processing field makes extensive use of syntactic parsers which assign structures in a meaning-free way. This annotation has been shown to be a useful stage in a processing pipeline which may or may not be followed by a semantic processing stage, depending on the application. To the extent that parses, like those that have been developed for English, are useful, we would like such tools for other languages. For the small number of languages for which we have treebanks available, supervised parsing techniques can be applied. However, the vast majority of languages have no treebank resources, and an unsupervised parser based on grammar induction techniques is the only alternative to the allocation of human expert resources.

Finally, as a matter of scope, the syntax-only learning task is a good way to further the understanding of how unsupervised inductive methods might effectively learn components of natural language.

1.2.3 Problematic Evaluation

A serious issue for the present work, as for all grammar induction systems, is evaluation. Such systems can be thought of as producing two kinds of output. First, they can be seen in the classical grammar induction view, where the result of the learning process is a grammar from some grammar family. When the target grammar is known, one can evaluate the degree to which the hypothesized grammar resembles the target. However, except for toy

experiments with grammar recovery (where one authors a grammar, generates from that grammar, then attempts to recover the generating grammar), we do not necessarily know the target grammar. Additionally, we may not have a satisfactory way to quantify the closeness of a hypothesis to a target grammar. Moreover, various systems may learn grammars from different grammar families. One option for evaluating learned grammars, which we will apply in this work, is to qualitatively evaluate them by inspection. This can be highly illuminating, but is unsatisfying on its own. Another option, which will also be used in this work, is to compare the tree structures predicted by the model to gold-standard trees produced by a linguist. While this measure is not itself subjective, the gold-standard is open to criticism. The issues and metrics of evaluation will be discussed in more depth in section 2.2.1.

1.3 Related Work

The work most closely related to this thesis can be broken into several types. A good deal of classical grammar induction work operated in a primarily symbolic fashion, learning symbolic context-free grammars, often predating the prevalence of probabilistic context-free grammars. Examples include Olivier (1968), Wolff (1988), *inter alia*. These methods will be discussed in section 4.3. More recent work, at least in the NLP community, has tended to embrace parameter search methods, usually using the Expectation-Maximization algorithm (EM) to fit probabilistic models to the data. These methods will be discussed in section 5.1 and section 6.1.2.

Chapter 2

Experimental Setup and Baselines

This chapter details the data sets used and the evaluation metrics reported in later chapters.

2.1 Input Corpora

The systems presented in this work all take as input a collection of sentences, where each word of the sentence is tagged with a word classes. The induction algorithms in this work are sensitive only to the word classes, not to the individual words. In all cases, sentences are taken from treebanks, which contain both sentences and their phrase-structure parses. The treebank parses are not used to guide the induction, but rather are used as a gold standard to evaluate the induction. The preterminal part-of-speech symbols in the treebank parses can be used as word classes, but need not be. We will describe here the general data pre-processing used in the context of the English Penn treebank, then briefly describe the differences for other languages and treebanks.

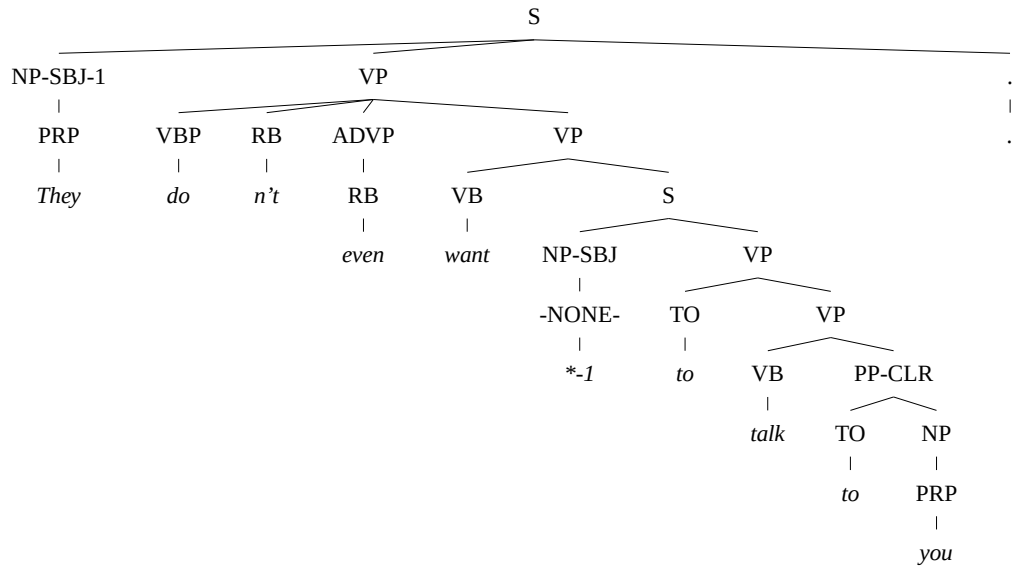
2.1.1 English data

For experiments on English, the treebank used is the Wall Street Journal (WSJ) section of the English Penn treebank (Marcus et al. 1993). This corpus is written English newswire, clearly not representative of the language child language learners are usually exposed to, but typical of the language generally parsed by supervised parsing systems.

One of the trees in this corpus is

They don't even want to talk to you.

and its treebank entry is



Some effort, described below, was made to alter this data to better represent the data available to a human learner, for example by removing empty elements and punctuation.

The example here contains the empty element **-1*, an empty marker indicating a controlled subject of the lower S. Empty elements in the English treebank can be identified by the reserved tag -NONE-. All tree nodes dominating no overt elements were pruned from the tree. Punctuation was similarly pruned, although it is arguable that at least some punctuation is correlated with information in an acoustic input, e.g. prosody. Words were considered to be punctuation whenever they were tagged with the following parts-of-speech (again, for English):

, . : “ ” -LRB- -RRB-

The final change to the yields of the English trees is that the tags $ and # were deleted, not because their contents are not pronounced, but because they are not pronounced where the tag occurs. This decision was comparatively arbitrary, but the results are little changed by leaving these items in. There are other broad differences between this data and spoken

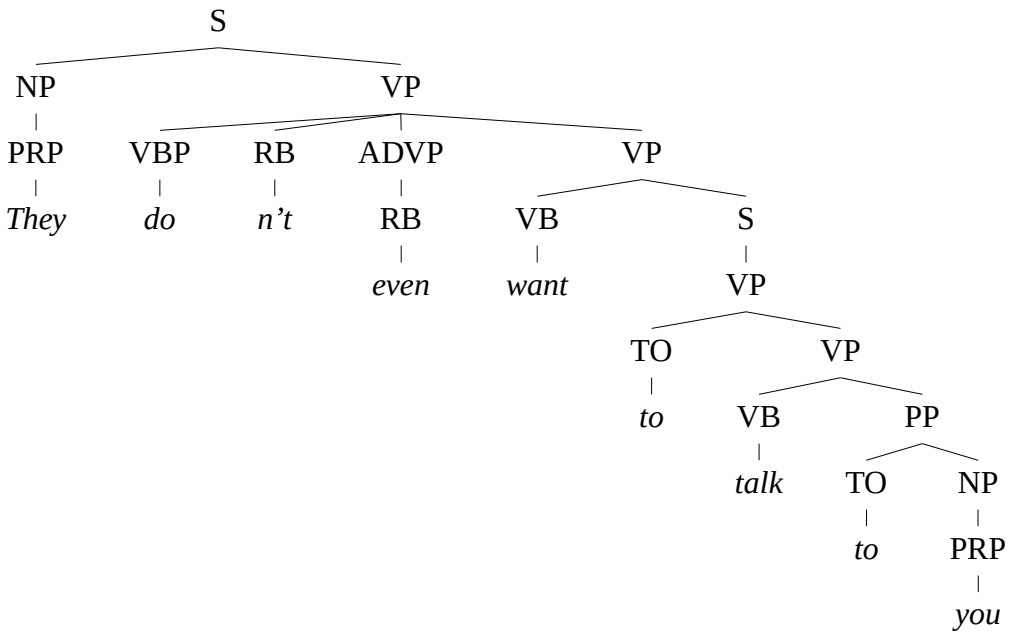


Figure 2.1: A processed gold tree, without punctuation, empty categories, or functional labels for the sentence, "They don't even want to talk to you."

language, which we make no attempt to alter. For example, it has already been tokenized in a very particular way, including that here *don't* has been split into *do* and *n't*. We leave this tokenization as it is.

Of course, treebank data is dissimilar to general spoken language in a number of ways, but we made only these few broad corrections; no spelling-out of numbers or re-ordering of words was done. Such spelling-out and re-ordering was done in Roark (2001), and could presumably be of benefit here, as well.[1]

For the trees themselves, which in fully unsupervised systems are used only for evaluation purposes, we always removed the functional tags from internal nodes. In this example, the final form of the tree would be as shown in figure 2.1.

From these trees, we extract the *preterminal yield*, consisting of the part-of-speech sequence. In this example, the preterminal yield is

[1]An obvious way to work with input more representative of spoken text would have been to use the Switchboard section of the the Penn Treebank, rather than the WSJ section. However, the Switchboard section has added complexities, such as dysfluencies and restarts, which, though clearly present in a child's language acquisition experience, complicate the modeling process.

PRP VBP RB RB VB TO VB TO PRP

From the English treebank, we formed two data sets. First, WSJ consists of the preterminal yields for all trees in the English Penn treebank. Second, WSJ10 consists of all preterminal yields of length at most 10 (length measured after the removals mentioned above). WSJ has 49208 trees, while WSJ10 has 7422 trees. Several experiments also refer to cutoffs less than or more than 10.

For several experiments, we also used the ATIS section of the English Penn treebank. The resulting sentences will be referred to as ATIS.

Data sets were similarly constructed from other corpora for experiments on other languages.

2.1.2 Chinese data

For our Chinese experiments, the Penn Chinese treebank (version 3) was used (Xue et al. 2002). The only tags removed were the empty-element tag -NONE- and the punctuation tag PU.[2] The set of at most 10 word sentences from this corpus will be referred to as CTB10 (2473 sentences).

2.1.3 German data

For our German language experiments, the NEGRA corpus was used (Skut et al. 1998). This corpus contains substantially more kinds of annotation than the Penn treebank, but we used the supplied translation of the corpus into Penn treebank-style tree structures. For the German data, we removed only the punctuation tags ($. $, $*LRB* $*RRB*) and empty-element tags (tags starting with *). The set of at most 10 word sentences from this corpus will be referred to as NEGRA10 (2175 sentences).

2.1.4 Automatically induced word classes

For the English data, we also constructed a variant of the English Penn treebank where the given part-of-speech preterminals were replaced with automatically-induced word classes.

[2]For some experiments, the punctuation tag was left in; these cases will be mentioned as they arise.

To induce these tags, we used the simplest method of (Schütze 1995) (which is close to the methods of (Schütze 1993, Finch 1993)). For (all-lowercased) word types in the Penn treebank, a 1000 element vector was made by counting how often each co-occurred with each of the 500 most common words immediately to the left or right in both the Treebank text and additional 1994–96 WSJ newswire. These vectors were length-normalized, and then rank-reduced by an SVD, keeping the 50 largest singular vectors. The resulting vectors were clustered into 200 word classes by a weighted k-means algorithm, and then grammar induction operated over these classes. We do not believe that the quality of our tags matches that of the better methods of Schütze (1995), much less the recent results of Clark (2000).

2.2 Evaluation

Evaluation of unsupervised methods is difficult in several ways. First, the evaluation objective is unclear, and will vary according to the motivation for the grammar induction.

If our aim is to produce a probabilistic language model, we will want to evaluate the grammar based on a density measure like perplexity of observed strings.

If our aim is to annotate sentences with syntactic markings which are intended to facilitate further processing, e.g. semantic analysis or information extraction, then we will want a way to measure how consistently the learned grammar marks its annotations, and how useful those annotations are to further processing. This goal would suggest a task-based evaluation, for example, turning the learned structures into features for other systems.

If our aim is essentially to automate the job of the linguist, then we will want to judge the learned grammar by whether they describe the structure of language in the way a linguist would. Of course, with many linguists comes many ideas of what the true grammar of a language is, but, setting this aside, we might compare the learned grammar to a reference grammar or grammars using some metric of grammar similarity.

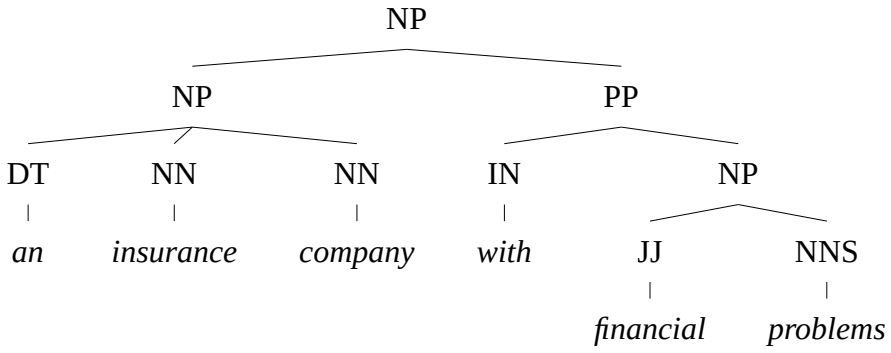
In this work, we take a stance in between the latter two desiderata, and compare the learned tree structures to a treebank of linguistically motivated gold-standard trees. To the extent that the gold standard is broadly representative of a linguistically correct grammar, systematic agreement with gold standard structures will indicate linguistic correctness of the learned models. Moreover, to the extent that the gold standard annotations have been

proven useful in further processing, matching the gold structures can reasonably be expected to correlate well with functional utility of the induced structures.

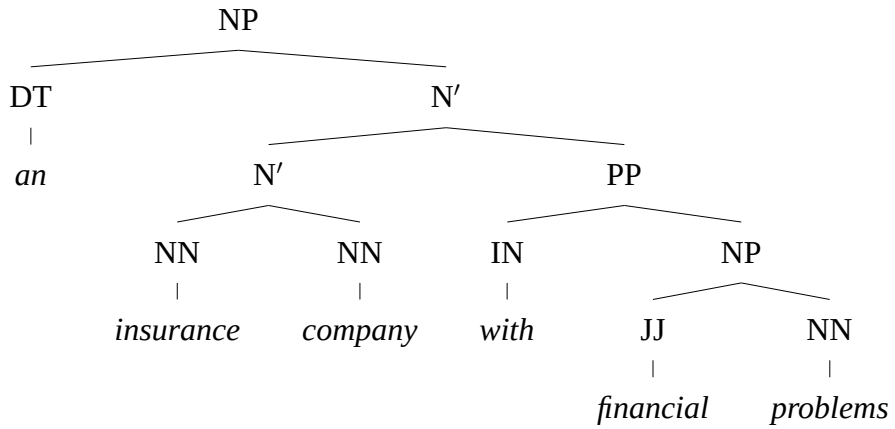
The approach of measuring agreement with the gold treebank – unsupervised parsing accuracy – is certainly not without its share of problems, as we describe in section 2.2. Most seriously, grammar induction systems often learn systematic alternate analyses of common phenomena, which can be devastating to basic bracket-match metrics. Despite these issues, beginning in section 2.2.1, we describe metrics for measuring agreement between induced trees and a gold treebank. Comparing two trees is a better-understood and better-established process than comparing two grammars, and by comparing hypothesized trees one can compare two systems which do not use the same grammar family, and even compare probabilistic and symbolic learners. Moreover, comparison of posited structures is the mechanism used by both work on supervised parsing and much previous work on grammar induction.

2.2.1 Alternate Analyses

There is a severe liability to evaluating a grammar induction system by comparing induced trees to human-annotated treebank trees: for many syntactic constructions, the syntactic analysis is debatable. For example, the English Penn Treebank analyzes *an insurance company with financial problems* as



while many linguists would argue for a structure more like



Here, the prepositional phrase is inside the scope of the determiner, and the noun phrase has at least some internal structure other than the PP (many linguists would want even more). However, the latter structure would score badly against the former: the N′ nodes, though reasonable or even superior, are either over-articulations (precision errors) or crossing brackets (both precision and recall errors). When our systems propose alternate analyses along these lines, it will be noted, but in any case it complicates the process of automatic evaluation.

To be clear what the dangers are, it is worth pointing out that a system which produced both analyses above in free variation would score better than one which only produced the latter. However, like choosing which side of the road to drive on, either convention is preferable to inconsistency.

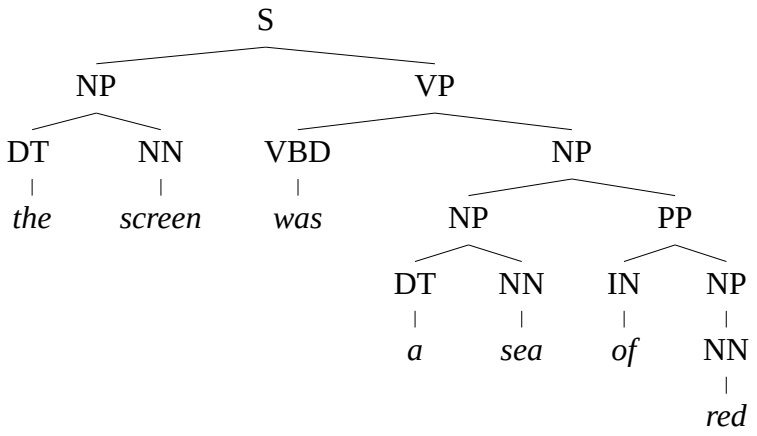
While there are issues with measuring parsing accuracy against gold standard treebanks, it has the substantial advantage of providing hard empirical numbers, and is therefore an important evaluation tool. We now discuss the specific metrics used in this work.

2.2.2 Unlabeled Brackets

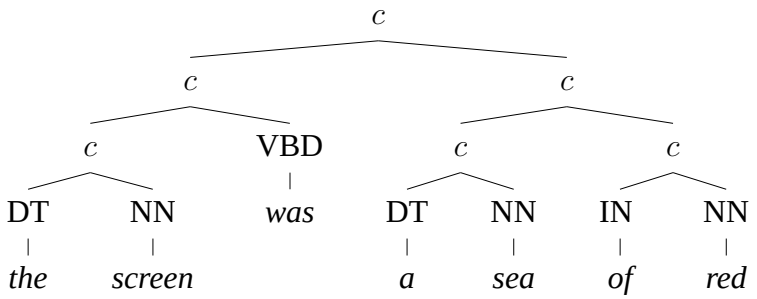
Consider the pair of parse trees shown in figure 2.2 for the sentence

$$_0\ \textit{the}\ _1\ \textit{screen}\ _2\ \textit{was}\ _3\ \textit{a}\ _4\ \textit{sea}\ _5\ \textit{of}\ _6\ \textit{red}\ _7$$

The tree in figure 2.2(a) is the gold standard tree from the Penn treebank, while the tree in figure 2.2(b) is an example output of a version of the induction system described in chapter 5. This system doesn't actually label the brackets in the tree; it just produces a



(a) Gold Tree



(b) Predicted Tree

Figure 2.2: A predicted tree and a gold treebank tree for the sentence, “The screen was a sea of red.”

nested set of brackets. Moreover, for systems which do label brackets, there is the problem of knowing how to match up induced symbols with gold symbols. This latter problem is the same issue which arises in evaluating clusterings, where cluster labels have no inherent link to true class labels. We avoid both the no-label and label-correspondence problems by measuring the unlabeled brackets only.

Formally, we consider a labeled tree T to be a set of labeled constituent brackets, one for each node n in the tree, of the form $(x : i, j)$, where x is the label of n, i is the index of the left extent of the material dominated by n, and j is the index of the right extent of the material dominated by n. Terminal and preterminal nodes are excluded, *as are non-terminal nodes which dominate only a single terminal*. For example, the gold tree (a) consists of the labeled brackets:

Constituent	Material Spanned
$(\text{NP} : 0, 2)$	*the screen*
$(\text{NP} : 3, 5)$	*a sea*
$(\text{PP} : 5, 7)$	*of red*
$(\text{NP} : 3, 7)$	*a sea of red*
$(\text{VP} : 2, 7)$	*was a sea of red*
$(\text{S} : 0, 7)$	*the screen was a sea of red*

From this set of labeled brackets, we can define the corresponding set of unlabeled brackets:

$$brackets(T) = \{\langle i, j\rangle : \exists x \text{ s.t. } (\text{X} : i, j) \in T\}$$

Note that even if there are multiple labeled constituents over a given span, there will be only a single unlabeled bracket in this set for that span. The definitions of unlabeled precision (UP) and recall (UR) of a proposed corpus $P = [P_i]$ against a gold corpus $G = [G_i]$ are:

$$\text{UP}(P, G) \equiv \frac{\sum_i |brackets(P_i) \cap brackets(G_i)|}{\sum_i |brackets(P_i)|}$$

$$\text{UR}(P, G) \equiv \frac{\sum_i |brackets(P_i) \cap brackets(G_i)|}{\sum_i |brackets(G_i)|}$$

In the example above, both trees have 6 brackets, with one mistake in each direction, giving a precision and recall of 5/6.

As a synthesis of these two quantities, we also report unlabeled F_1, their harmonic mean:

$$\mathrm{UF}_1(P, G) = \frac{2}{\mathrm{UP}(P, G)^{-1} + \mathrm{UR}(P, G)^{-1}}$$

Note that these measures differ from the standard PARSEVAL measures (Black et al. 1991) over pairs of labeled corpora in several ways: multiplicity of brackets is ignored, brackets of span one are ignored, and bracket labels are ignored.

2.2.3 Crossing Brackets and Non-Crossing Recall

Consider the pair of trees in figure 2.3(a) and (b), for the sentence *a full four-color page in newsweek will cost 100,980*.[3]

There are several precision errors: due the flatness of the gold treebank, the analysis inside the NP *a full four-color page* creates two incorrect brackets in the proposed tree. However, these two brackets do not actually cross, or contradict, any brackets in the gold tree. On the other hand, the bracket over the verb group *will cost* does contradict the gold tree's VP node. Therefore, we define several additional measures which count as mistakes only contradictory brackets. We write $b \sim S$ for an unlabeled bracket b and a set of unlabeled brackets S if b does not cross any bracket $b' \in S$, where two brackets are considered to be crossing if and only if they overlap but neither contains the other.

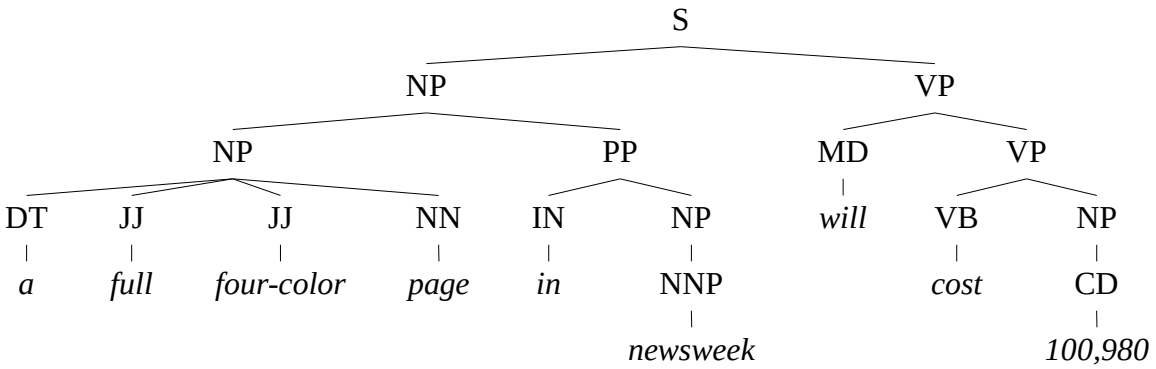
The definitions of unlabeled non-crossing precision (UNCP) and recall (UNCR) are

$$\mathrm{UNCP}(P, G) \equiv \frac{\sum_i |\{b \in brackets(P_i) : b \sim brackets(G_i)\}|}{\sum_i |brackets(P_i)|}$$

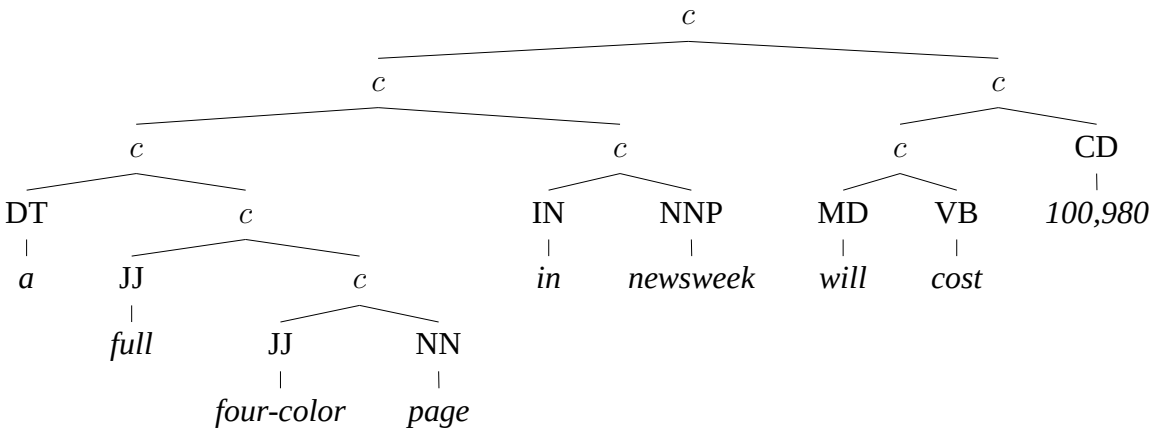
$$\mathrm{UNCR}(P, G) \equiv \frac{\sum_i |\{b \in brackets(G_i) \cap brackets(P_i)\}|}{\sum_i |brackets(G_i)|}$$

and unlabeled non-crossing F_1 is defined as their harmonic mean as usual. Note that these measures are more lenient than UP/UR/UF_1. Where the former metrics count all proposed analyses of structure inside underspecified gold structures as wrong, these measures count

[3] Note the removal of the *$* from what was originally *$ 100,980* here.



(a) Gold Tree



(b) Predicted Tree

Figure 2.3: A predicted tree and a gold treebank tree for the sentence, “A full, four-color page in Newsweek will cost $100,980.”

all such analyses as correct. The truth usually appears to be somewhere in between.

Another useful statistic is the *crossing brackets rate* (CB), the average number of guessed brackets per sentence which cross one or more gold brackets:

$$\mathrm{CB}(P,G) \equiv \frac{\sum_i |\{b \in brackets(T_{P_i}) : \neg b\, brackets(T_{G_i})\}|}{|P|}$$

2.2.4 Per-Category Unlabeled Recall

Although the proposed trees will either be unlabeled or have labels with no inherent link to gold treebank labels, we can still report a per-label recall rate for each label in the gold label vocabulary. For a gold label x, that category's labeled recall rate (LR) is defined as

$$\mathrm{LR}(x,P,G) \equiv \frac{\sum_i |(\mathrm{x} : i,j) \in G_i : j > i+1 \wedge \langle i,j\rangle \in brackets(P_i)|}{\sum_i |(\mathrm{x} : i,j) \in G_i|}$$

In other words, we take all occurrences of nodes labeled x in the gold treebank which dominate more than one terminal. Then, we count the fraction which, as unlabeled brackets, have a match in their corresponding proposed tree.

2.2.5 Alternate Unlabeled Bracket Measures

In some sections, we report results according to an alternate unlabeled bracket measure, which was originally used in earlier experiments. The alternate unlabeled bracket precision (UP′) and recall (UR′) are defined as

$$\mathrm{UP}'(P,G) \equiv \sum_i \frac{|brackets(P_i) \cap brackets(G_i)| - 1}{|brackets(P_i)| - 1}$$

$$\mathrm{UR}'(P,G) \equiv \sum_i \frac{|brackets(P_i) \cap brackets(G_i)| - 1}{|brackets(G_i)| - 1}$$

with, F_1 (UF_1') defined as their harmonic mean, as usual. In the rare cases where it occurred, a ratio of 0/0 was taken to be equal to 1. These alternate measures do not count the top bracket as a constituent, since, like span-one constituents, all well-formed trees contain the top bracket. This exclusion tended to lower the scores. On the other hand, the scores

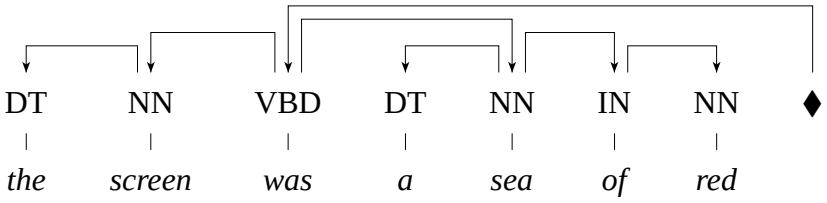
were macro-averaged at the sentence level, which tended to increase the scores. The net differences were generally fairly slight, but for the sake of continuity we report some older results by this metric.

2.2.6 EVALB

For comparison to earlier work which tested on the ATIS corpus using the EVALB program with the supplied unlabeled evaluation settings, we report (though only once) the results of running our predicted and gold versions of the ATIS sentences through EVALB (see section 5.3. The difference between the measures above and the EVALB program is that the program has a complex treatment of multiplicity of brackets, while the measures above simply ignore multiplicity.

2.2.7 Dependency Accuracy

The models in chapter 6 model dependencies, linkages between pairs of words, rather than top-down nested structures (although certain equivalences do exist between the two representations, see section 6.1.1). In this setting, we view trees not as collections of constituent brackets, but rather as sets of word pairs. The general meaning of a dependency pair is that one word either modifies or predicates the other word. For example, in *the screen was a sea of red*, we get the following dependency structure:



Arrows denote dependencies. When there is an arrow from a (tagged) word w_h to another word (tagged) w_a, we say that w_h is the *head* of the dependency, while we say that w_a is the *argument* of the dependency. Unlabeled dependencies like those shown conflate various kinds of relationships that can exist between words, such as modification, predication, and delimitation, into a single generic one, in much the same way as unlabeled

brackets collapse the distinctions between various kinds of constituents.[4] The dependency graphs we consider in this work are all tree-structured, with a reserved *root* symbol ♦ at the head of the tree, which always has exactly one argument (the head of the sentence); that link forms the *root dependency*.

All dependency structures for a sentence of n words (not counting the root) will have n dependencies (counting the root dependency). Therefore, we can measure dependency accuracy straightforwardly by comparing the dependencies in a proposed corpus against those in a gold corpus. There are two variations on dependency accuracy in this work: directed and undirected accuracy. In the directed case, a proposed word pair is correct only if it is in the gold parse in the same direction. For the undirected case, the order is ignored. Note that two structures which agree exactly as undirected structures will also agree as directed structures, since the root induces a unique head-outward ordering over all other dependencies.

One serious issue with measuring dependency accuracy is that, for the data sets above, the only human-labeled head information appears in certain locations in the NEGRA corpus. However, for these corpora, gold phrase structure can be heuristically transduced to gold dependency structures with reasonable accuracy using rules such as in Collins (1999). These rules are imperfect (for example, in *new york stock exchange lawyers*, the word *lawyers* is correctly taken to be the head, but each of the other words links directly to it, incorrectly for *new, york,* and *stock*). However, for the moment it seems that the accuracy of unsupervised systems can still be meaningfully compared to this low-carat standard. Nonetheless, we discuss these issues more when evaluating dependency induction systems.

2.3 Baselines and Bounds

In order to meaningfully measure the performance of our systems, it is useful to have baselines, as well as upper bounds, to situate accuracy figures. We describe these baselines and bounds below; figures of their performance will be mentioned as appropriate in later chapters.

[4]The most severe oversimplification is not any of these collapses, but rather the treatment of conjunctions, which do not fit neatly into this word-to-word linkage framework.

2.3.1 Constituency Trees

The constituency trees produced by the systems in this work are all usually binary branching. The trees in the gold treebanks, such as in figure 2.1, are, in general, not. Gold trees may have unary productions because of singleton constructions or removed empty elements (for example, figure 2.1). Gold trees may also have ternary or flatter productions, either because such constructions seem correct (for example in coordination structures) or because the treebank annotation standards left certain structures intentionally flat (for example, inside noun phrases, figure 2.3(a)).

Upper Bound on Precision

Unary productions are not much of a concern, since their presence does not change the set of unlabeled brackets used in the measures of the previous section. However, when the proposed trees are more articulated that the gold trees, the general result will be a system which exhibits higher recall than precision. Moreover, for gold trees which have nodes with three or more children, it will be impossible to achieve a perfect precision. Therefore, against any treebank which has ternary or flatter nodes, there will be an upper bound on the precision achievable by a system which produces binary trees only.

Random Trees

A minimum standard for an unsupervised system to claim a degree of success is that it produce parses which are of higher quality than selecting parse trees at random from some uninformed distribution.[5] For the random baseline in this work, we used the uniform distribution over binary trees. That is, given a sentence of length n, all distinct unlabeled trees over n items were given equal weight. This definition is procedurally equivalent to parsing with a grammar which has only one nonterminal production $x \rightarrow x \quad x$ with weight 1. To get the parsing scores according to this random parser, one can either sample a parse or parses at random or calculate the expected value of the score. Except where noted otherwise, we did the latter; see appendix B.1 for details on computing the posteriors of this

[5] Again, it is worth pointing out that a below-random match to the gold treebank may indicate a good parser if there is something seriously wrong or arbitrary about the gold treebank.

distribution, which can be done in closed form.

Left- and Right-Branching Trees

Choosing the entirely left- or right-branching structure ($B = \{\langle 0, i\rangle : i \in [1, n]\}$ or $B = \{\langle i, n\rangle : i \in [0, n-1]\}$, respectively, see figure 2.4) over a test sentence is certainly an uniformed baseline in the sense that the posited structure for a sentence independent of any details of that sentence save its length. For English, right-branching structure happens to be an astonishingly good baseline. However, it would be unlikely to perform well for a VOS language like Malagasy or VSO languages like Hebrew; it certainly is not nearly so strong for the German and Chinese corpora tested in this work. Moreover, the knowledge that right-branching structure is better for English than left-branching structure is a language-specific bias, even if only a minimal one. Therefore, while our systems do exceed these baselines, that has not always been true for unsupervised systems which had valid claims of interesting learned structure.

2.3.2 Dependency Baselines

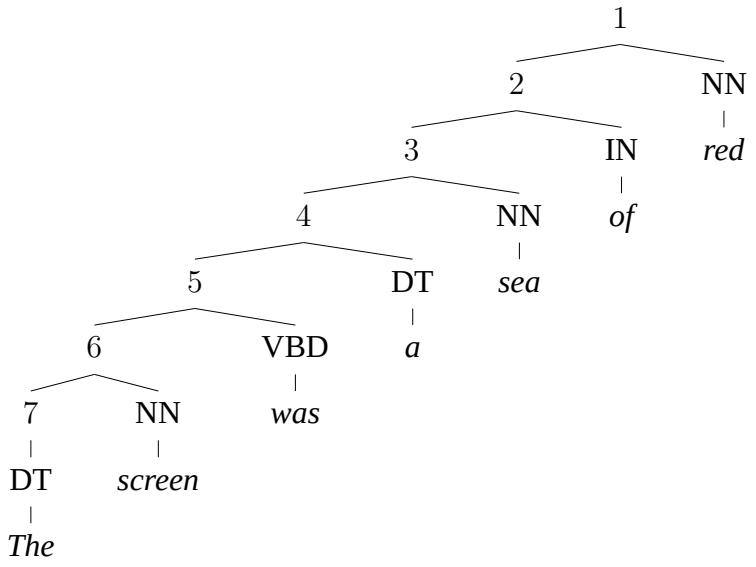
For dependency tree structures, all n word sentences have n dependencies, including the root dependency. Therefore, there is no systematic upper bound on achievable dependency accuracy. There are sensible baselines, however.

Random Trees

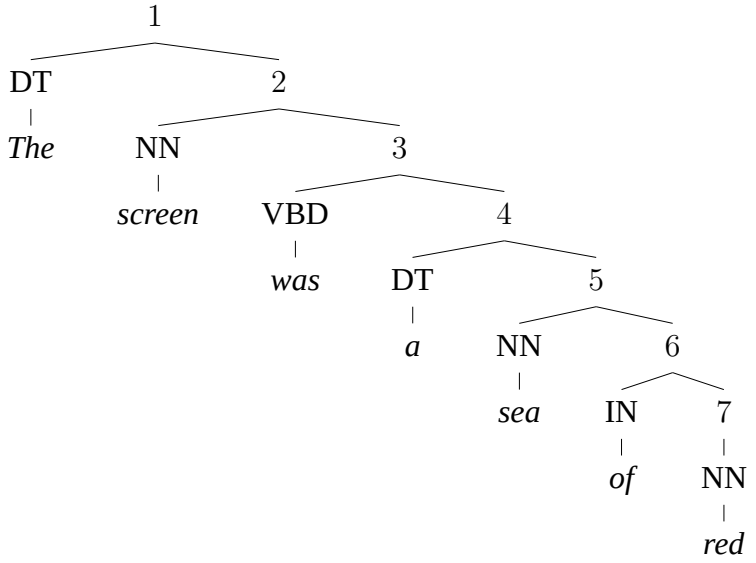
Similarly to the constituency tree case, we can get a lower bound by choosing dependency trees at random. In this case, we extracted random trees by running the dependency model of chapter 6 with all local model scores equal to 1.

Adjacent Links

Perhaps surprisingly, most dependencies in natural languages are between adjacent words, for example nouns and adjacent adjectives or verbs and adjacent adverbs. This actually suggests two baselines, shown in figure 2.5. In the backward adjacent baseline, each word

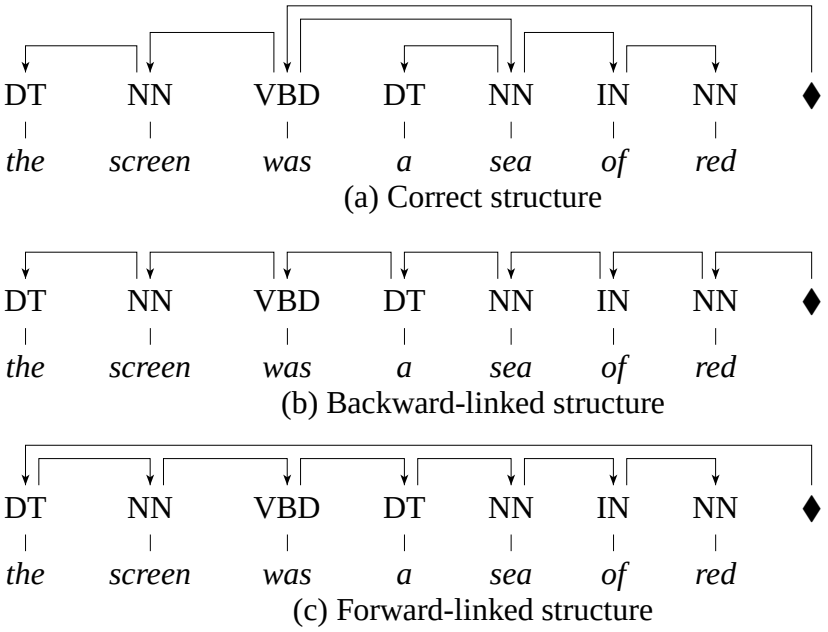


(a) Left-branching structure



(b) Right-branching structure

Figure 2.4: Left-branching and right-branching baselines.



(a) Correct structure

(b) Backward-linked structure

(c) Forward-linked structure

Figure 2.5: Adjacent-link dependency baselines.

takes the word before it as an argument, with the last word of the sentence being the head of the sentence (i.e., linked to the root). This direction corresponds to left-branching constituent structure. The forward adjacent baseline is the result of making the first word the head of the sentence and having each word be the argument of the preceding word. This direction corresponds to right-branching constituent structure. While it is true that most dependencies are local, it is not true that they are overwhelmingly leftward or rightward in direction; the adjacent-link baseline is therefore most competitive on the undirected dependency accuracy measure.

Chapter 3

Distributional Methods

One area of language learning which has seen substantial success is the task of inducing word classes, such as parts-of-speech and semantic fields. This success is largely due to simple, robust distributional methods, which we define and examine in this chapter. The basic distributional approach for word classes can be used in several ways to drive tree induction; we present a more traditional structure-search method in chapter 4 and a much more successful parameter search approach in chapter 5.

3.1 Parts-of-speech and Interchangeability

The linguistic notion of a part-of-speech is motivated by the fact that there are large sets of words which are syntactically interchangeable. For example, we have

(a) *The cat went **over** the box.*
(b) *The cat went **under** the box.*
(c) *The cat went **inside** the box.*
(d) ??*The cat went **among** the box.*
(d) **The cat went **of** the box.*
(d) **The cat went **the** the box.*

There is class of words, including *over, under,* and *inside,* which can be substituted between the verb and noun phrase in the sentences, changing the meaning but preserving the syntax.

This class is roughly the set of prepositions, though not all linguistic prepositions can occur equally well here. In particular, *of* is usually taken to be a preposition, but usually heads noun- and adjective-modifying prepositional phrases, and cannot occur here. The preposition *among* is inappropriate, since it places a mass or plurality requirement on its object. Nonetheless, a sufficiently large collection of examples in which the set of prepositions are generally mutually interchangeable can be used to motivate the coherence of prepositions as a part-of-speech, with finer details distinguishing various subclasses.

3.2 Contexts and Context Distributions

So what does mutual substitutability mean for a learning system? The strong claim behind the part-of-speech level is that the syntactic behavior of a word depends only on the part-of-speech of that word, at least broadly speaking.[1] Therefore, we should be able to collect data about which contexts various words occur in, and use this information to detect parts-of-speech.

The first operational question is how to tell what context a word is in. A particularly simple definition is to say that the context of a word is the pair of words immediately adjacent to the left and right. For example, in the sentence *the cat went over the box*, the word *over* occurs in the context $\langle$*went* – *the*$\rangle$. This is the *local linear context*, and has the advantage of being easy to identify (Finch and Chater 1992). Moreover, it is a reasonable hypothesis that the linear context is sufficient for inducing syntactic classes (cf. discussion of semantic classes below). Especially if one takes the view that linear context will be strongly correlated with other notions of context, it is an empirical issue if and how this

[1]This claim is often taken to be at odds with the success of lexicalization in syntactic modeling (Collins 1999, Charniak 2000) – if we actually need to know what words are in the sentence to parse well, doesn't that mean there's more to the syntax of a word than the part-of-speech indicates? However, there are three issues here. First, linguists are generally claiming that parts-of-speech suffice for describing which words can grammatically be substituted, not which words actually are substituted empirically, so there is no statistical independence claim in linguistic argumentation. Second, linguists consider parts-of-speech at various granularities: nouns, mass nouns, feminine mass nouns, etc. Finer levels influence finer syntactic phenomena. The broadest levels of nouns, verbs, and adjectives are intended to describe the broadest syntactic phenomena. So it should not be surprising that knowing that a word is a noun is less useful than knowing it is the noun *stocks*, but there are levels in between. Finally, disambiguation is partially semantic, and at that level parts-of-speech are not intended to reflect interchangeability.

context might be practically inadequate.

For linguistic argumentation, we generally use more articulated notions of context. For example, consider the following the sentences.

(a) *The cat went **over** the box.*

(b) *The cat jumps **over** people's feet.*

(a) *The cat thinks **that/*over** the box will support it.*

Both (a) and (b) are examples of *over* occurring in roughly the same context (between a verb and a noun phrase), while (c) is an example of a different context (between a verb and a subordinate clause), despite the fact that the local linear context is in fact more similar between (a) and (c). In linguistic argumentation, we actually present the entire picture of a grammar all at once, and essentially argue for its coherence, which is problematic for automated methods, and most work on part-of-speech learning has used the local linear context. We will discuss a hierarchical definition of context in section 3.4, but for now we consider surface contexts.

A few more notes about contexts. In the local linear context, each occurrence of a word corresponds to a single context (the adjacent word pair). However, we can consider that context to be either atomic or structured (joint or factored). In the structured view, we might break $\langle$*went – the*$\rangle$ down into multiple *context events*. For example, Finch and Chater (1992) and Schütze (1995) break these contexts into a left event $\langle$*went* – $\rangle$ and a right event $\langle$ – *the*$\rangle$. For non-local contexts, this decomposition is virtually obligatory. Consider the *100-word linear context*, consisting of the 100-words on either side of the target. In this case, the context might be decomposed into 200 position- and direction-free events, resulting in the bag of words inside that window.

3.3 Distributional Word-Classes

Formally, if w is a word from some vocabulary W, let $\sigma(w)$, called the *signature* of w, denote the counts of each context event in some training corpus, with context events ranging over a vocabulary X. There is a great deal of work which uses such signatures to detect word classes; we discuss only a few examples here.

The general approach of distributional clustering is to view the data as a $|W| \times |X|$ matrix M, where there is a row for each word w, and a column for each context event type x. Equivalently, one can think of M as the result of stacking together all the signatures $\sigma(w)$. Here, work varies, but most approaches attempt to find a low dimensional representation of M.

The basic method of Schütze (1995) uses the decomposed local linear context (instead of joint counts) and restricts context events to the most frequent n words. He then row-normalizes M (so the rows are probability distributions over those context events), and uses a truncated singular-value decomposition to write $M = U\Sigma V'$, where the top r left eigenvectors (columns in U) are retained. U is then a $|W| \times r$ matrix, with each column representing a component weight in one of r latent dimensions. The rows of U are clustered using the k-means algorithm into flat clusters. This approach operates under the assumption that M is a low-rank matrix distorted by Gaussian noise.[2] This brief sketch of Schütze's paper omits the crucial innovation of that work, which was a mechanism for contextually disambiguating words which belong to multiple classes, which is crucial if one is hoping to reconstruct word classes that look like traditional parts-of-speech.[3]

Since the signatures have a probabilistic interpretation, it is reasonable to think of M as an empirical sample from a joint probability over words and their contexts. This kind of approach is taken in Pereira et al. (1993), *inter alia*. Here, we think of M as having been sampled from $\mathrm{P}(W, X)$. We assume a hidden class variable and write

$$\mathrm{P}(W, X) = \mathrm{P}(C)\mathrm{P}(W|C)\mathrm{P}(X|C)$$

We then try to find estimates which maximize the likelihood of M, either using EM or specialized cluster-reassignment algorithms (Clark 2000). The appeal of using a probabilistic divergence instead of least-squares is somewhat offset by the fact that not only are the independence assumptions in the latent model false (as always), the samples aren't even IID – one word's left context is another word's right context.

[2] We used this method to produce our own word-clusters in some experiments; in those cases, we used $r = 100$ and $k = 200$.

[3] Without the ability to represent ambiguous words as mixtures of multiple simple classes, words such as *to*, which can be either a preposition or an infinitive marker, show up as belonging to completely separate classes which represent that ambiguity mixture.

It's worth pointing out that if one considers the context to be the entire document and uses the position- and direction-free bag-of-words decomposition into context events, these two sketched approaches correspond to LSA (Landauer et al. 1998) (modulo normalizations) and PLSA (Hofmann 1999), with words rather than documents as the row indices. One still gets word classes out of such contexts; they're just semantic or topical classes rather than the more syntactic classes produced by local contexts.

Again, there has been a lot of work in this area, much of it providing substantial extensions to the above methods. Two particularly interesting and successful extensions are presented in Clark (2000) and Clark (2003). The latter employs a model of $\mathrm{P}(W|C)$ in which words, rather than being generated as opaque symbols, are generated with internal character/morpheme-level structure. Thus, there is pressure for words which share suffixes, for example, to be put into the same class. The innovation presented in Clark (2000) is that, rather than consider the context of a word to be the adjacent words (as in Finch and Chater (1992)), or the classes of the adjacent words according to a preliminary clustering (as in Schütze (1995)), he considers it to be the classes according to the current model. This definition is circular, since the word classes are exactly what's being learned, and so there is an iterative process of reclustering followed by signature refinement.

To raise a point that will be revisited in chapter 5, one can compare the context clustering approaches above with HMM induction. After describing the Clark (2000) work, it might seem obvious that learning an HMM is the "correct" way of learning a model in which words are fully mediated by their classes and a word's class interacts directly with the preceding and following classes. There are at least three reasons, one practical, one empirical, and one conceptual, why there is a healthy amount of successful work on local clustering, but not on HMM induction. The practical reason is that signatures are often built using massive amounts of text, e.g. years of newswire. The full word-signature matrix is far too big to store, and so only counts over frequent contexts are retained. This shortcut is easier to work into a local clustering approach. The empirical reason is that, while even very early attempts at distributional clustering provided reasonably good word classes (Finch and Chater 1992), early attempts at inducing HMMs were disappointing, even when highly constrained (Merialdo 1994). Our experience with learning HMMs with EM suggests a conceptual explanation for these findings. Because word classes are a highly local

kind of syntax – local in the sense that they are not intended to encode any sentence- or phrase-wide syntax. However, HMMs are most certainly capable of encoding global state, such as whether the sentence has many numbers or many capitalized words. Such global indicators can be multiplied into the state space, resulting in strange learned classes. For these reasons, a model which does not properly represent a global structure can actually learn better local classes, at the possible cost of feeling less aesthetically satisfying.

3.4 Distributional Syntax

Distributional methods can be applied above the word level. For example, we can consider sequences of word classes, such as the part-of-speech sequences in a tagged corpus.

Figure 3.1 shows the most frequent local linear contexts for the parts-of-speech occurring in the WSJ corpus. These signatures encapsulate much of the broad linear syntactic trends of English, in essentially the same way a markov model over tag sequences would. For example, determiners generally precede nouns and follow prepositions, verbs, and sentence boundaries, just as one would expect. What is additionally useful about these signatures, and what is implicitly used in word-clustering approaches, is that similarity between local linear signatures correlates with syntactic relatedness. Figure 3.2 shows the top pairs of treebank part-of-speech tags, sorted by the Jensen-Shannon divergence between the two tags' signatures:

$$D_{JS}(p,q) = \frac{1}{2} D_{KL}(p|\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q|\frac{p+q}{2})$$

where D_{KL} is the Kullback-Leibler divergence:

$$D_{KL}(p|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

The lowest divergence (highest similarity) pairs are primarily of two types. First, there are pairs like $\langle \mathrm{VBD}, \mathrm{VBZ} \rangle$ (past vs. 3sg present tense finite verbs) and $\langle \mathrm{NN}, \mathrm{NNS} \rangle$ (singular vs. plural common nouns) where the original distinction was morphological, with minimal distributional reflexes. Second, there are pairs like $\langle \mathrm{DT}, \mathrm{PRP\$} \rangle$ (determiners vs. possessive

Tag	Top Linear Contexts by Frequency
CC	⟨NNP – NNP⟩, ⟨NN – NN⟩, ⟨NNS – NNS⟩, ⟨CD – CD⟩, ⟨NN – JJ⟩
CD	⟨IN – CD⟩, ⟨CD – IN⟩, ⟨IN – NN⟩, ⟨IN – NNS⟩, ⟨TO – CD⟩
DT	⟨IN – NN⟩, ⟨IN – JJ⟩, ⟨IN – NNP⟩, ⟨◇ – NN⟩, ⟨VB – NN⟩
EX	⟨◇ – VBZ⟩, ⟨◇ – VBP⟩, ⟨IN – VBZ⟩, ⟨◇ – VBD⟩, ⟨CC – VBZ⟩
FW	⟨NNP – NNP⟩, ⟨NN – FW⟩, ⟨DT – FW⟩, ⟨FW – NN⟩, ⟨FW – FW⟩
IN	⟨NN – DT⟩, ⟨NN – NNP⟩, ⟨NNS – DT⟩, ⟨NN – NN⟩, ⟨NN – JJ⟩
JJR	⟨DT – NN⟩, ⟨IN – IN⟩, ⟨RB – IN⟩, ⟨IN – NNS⟩, ⟨VBD – IN⟩
JJS	⟨DT – NN⟩, ⟨IN – CD⟩, ⟨POS – NN⟩, ⟨DT – NNS⟩, ⟨DT – JJ⟩
JJ	⟨DT – NN⟩, ⟨IN – NNS⟩, ⟨IN – NN⟩, ⟨JJ – NN⟩, ⟨DT – NNS⟩
LS	⟨◇ – VB⟩, ⟨◇ – JJ⟩, ⟨◇ – IN⟩, ⟨◇ – NN⟩, ⟨◇ – PRP⟩
MD	⟨NN – VB⟩, ⟨PRP – VB⟩, ⟨NNS – VB⟩, ⟨NNP – VB⟩, ⟨WDT – VB⟩
NNPS	⟨NNP – NNP⟩, ⟨NNP – ◇⟩, ⟨NNP – VBD⟩, ⟨NNP – IN⟩, ⟨NNP – CC⟩
NNP	⟨NNP – NNP⟩, ⟨IN – NNP⟩, ⟨◇ – NNP⟩, ⟨NNP – VBD⟩, ⟨DT – NNP⟩
NNS	⟨JJ – IN⟩, ⟨JJ – ◇⟩, ⟨NN – IN⟩, ⟨IN – IN⟩, ⟨NN – ◇⟩
NN	⟨DT – IN⟩, ⟨JJ – IN⟩, ⟨DT – NN⟩, ⟨NN – IN⟩, ⟨JJ – ◇⟩
PDT	⟨IN – DT⟩, ⟨VB – DT⟩, ⟨◇ – DT⟩, ⟨RB – DT⟩, ⟨VBD – DT⟩
POS	⟨NNP – NN⟩, ⟨NN – NN⟩, ⟨NNP – JJ⟩, ⟨NNP – NNP⟩, ⟨NN – JJ⟩
PR$	⟨IN – NN⟩, ⟨IN – JJ⟩, ⟨IN – NNS⟩, ⟨VB – NN⟩, ⟨VBD – NN⟩
PRP	⟨◇ – VBD⟩, ⟨◇ – VBP⟩, ⟨◇ – VBZ⟩, ⟨IN – VBD⟩, ⟨VBD – VBD⟩
RBR	⟨NN – ◇⟩, ⟨DT – JJ⟩, ⟨RB – JJ⟩, ⟨RB – IN⟩, ⟨NN – IN⟩
RBS	⟨DT – JJ⟩, ⟨POS – JJ⟩, ⟨CC – JJ⟩, ⟨PR$ – JJ⟩, ⟨VBZ – JJ⟩
RB	⟨MD – VB⟩, ⟨NN – IN⟩, ⟨RB – IN⟩, ⟨VBZ – VBN⟩, ⟨VBZ – JJ⟩
RP	⟨VB – DT⟩, ⟨VBN – IN⟩, ⟨VBD – IN⟩, ⟨VB – IN⟩, ⟨VBD – DT⟩
SYM	⟨◇ – IN⟩, ⟨◇ – VBZ⟩, ⟨◇ – NN⟩, ⟨◇ – JJ⟩, ⟨◇ – VBN⟩
TO	⟨NN – VB⟩, ⟨NNS – VB⟩, ⟨VBN – VB⟩, ⟨VBD – VB⟩, ⟨JJ – VB⟩
UH	⟨◇ – PRP⟩, ⟨◇ – DT⟩, ⟨◇ – ◇⟩, ⟨◇ – UH⟩, ⟨UH – ◇⟩
VBD	⟨NNP – DT⟩, ⟨NN – DT⟩, ⟨NN – VBN⟩, ⟨NN – IN⟩, ⟨NNP – PRP⟩
VBG	⟨IN – DT⟩, ⟨NN – DT⟩, ⟨DT – NN⟩, ⟨IN – NNS⟩, ⟨IN – NN⟩
VBN	⟨NN – IN⟩, ⟨VBD – IN⟩, ⟨NNS – IN⟩, ⟨VB – IN⟩, ⟨RB – IN⟩
VBP	⟨NNS – VBN⟩, ⟨NNS – RB⟩, ⟨PRP – RB⟩, ⟨NNS – DT⟩, ⟨NNS – IN⟩
VBZ	⟨NN – VBN⟩, ⟨NN – RB⟩, ⟨NN – DT⟩, ⟨NNP – VBN⟩, ⟨NNP – DT⟩
VB	⟨TO – DT⟩, ⟨TO – IN⟩, ⟨MD – DT⟩, ⟨MD – VBN⟩, ⟨TO – JJ⟩
WDT	⟨NN – VBZ⟩, ⟨NNS – VBP⟩, ⟨NN – VBD⟩, ⟨NNP – VBZ⟩, ⟨NN – MD⟩
W$	⟨NNP – NN⟩, ⟨NNP – NNS⟩, ⟨NN – NN⟩, ⟨NNS – NNS⟩, ⟨NNP – JJ⟩
WP	⟨NNS – VBP⟩, ⟨NNP – VBD⟩, ⟨NNP – VBZ⟩, ⟨NNS – VBD⟩, ⟨NN – VBZ⟩
WRB	⟨NN – DT⟩, ⟨NN – PRP⟩, ⟨◇ – DT⟩, ⟨◇ – PRP⟩, ⟨NN – NNS⟩

Figure 3.1: The most frequent left/right tag context pairs for the part-of-speech tags in the Penn Treebank.

pronouns) and $\langle \mathrm{WDT}, \mathrm{WP} \rangle$ (*wh*-determiners vs. *wh*-pronouns) where the syntactic role is truly different at some deep level, but where the syntactic peculiarities of English prevent there from being an easily detected distributional reflex of that difference. For example, English noun phrases can begin with either a DT like *the* in *the general idea* or a PRP$ like *his* in *his general idea*. However, they both go in the same initial position and cannot co-occur.[4] However, in general, similar signatures do reflect broadly similar syntactic functions.

One might hope that this correlation between similarity of local linear context signatures and syntactic function would extend to units of longer length. For example, DT JJ NN and DT NN, both noun phrases, might be expected to have similar signatures. Figure 3.2 shows the top pairs of multi-tag subsequences by the same signature divergence metric.[5]

Of course, linguistic arguments of syntactic similarity involve more than linear context distributions. For one, traditional argumentation places more emphasis on potential substitutability (what contexts items *can* be used in) and less emphasis on empirical substitutability (what contexts they *are* used in) (Radford 1988). We might attempt to model this in some way, such as by flattening the empirical distribution over context counts to blunt the effects of non-uniform empirical usage of grammatical contexts. For example, we could use as our context signatures the distribution which is uniform over observed contexts, and zero elsewhere.

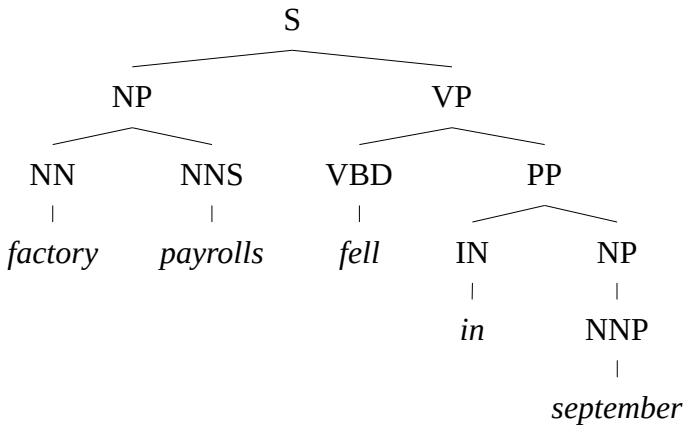
Another difference between linear context distributions and traditional linguistic notions of context is that traditional contexts refer to the surrounding high-level phrase structure. For example, the subsequence *factory payrolls* in the sentence below is, linearly, followed by *fell* (or, at the tag level, VBD). However, in the treebank parse

[4] Compare languages such as Italian where the PRP$ would require a preceding DT, as in *la sua idea*, and where this distributional similarity would not appear.

[5] The list of candidates was restricted to pairs of items each of length at most 4 and each occurring at least 50 times in the treebank – otherwise the top examples are mostly long singletons with chance zero divergence.

Rank	Tag Pairs	Sequence Pairs
1	⟨ VBZ, VBD ⟩	⟨ NNP NNP, NNP NNP NNP ⟩
2	⟨ DT, PRP$ ⟩	⟨ DT JJ NN IN, DT NN IN ⟩
3	⟨ NN, NNS ⟩	⟨ NNP NNP NNP NNP, NNP NNP NNP ⟩
4	⟨ WDT, WP ⟩	⟨ DT NNP NNP, DT NNP ⟩
5	⟨ VBG, VBN ⟩	⟨ IN DT JJ NN, IN DT NN ⟩
6	⟨ VBP, VBD ⟩	⟨ DT JJ NN, DT NN NN ⟩
7	⟨ VBP, VBZ ⟩	⟨ DT JJ NN, DT NN ⟩
8	⟨ EX, PRP ⟩	⟨ IN JJ NNS, IN NNS ⟩
9	⟨ POS, WP$ ⟩	⟨ IN NN IN, IN DT NN IN ⟩
10	⟨ RB, VBN ⟩	⟨ IN NN, IN JJ NN ⟩
11	⟨ CD, JJ ⟩	⟨ DT JJ NN NN, DT NN NN ⟩
12	⟨ NNPS, NNP ⟩	⟨ IN NNP, IN NNP NNP ⟩
13	⟨ CC, IN ⟩	⟨ IN JJ NNS, IN NN NNS ⟩
14	⟨ JJS, JJR ⟩	⟨ NN IN DT, NN DT ⟩
15	⟨ RB, VBG ⟩	⟨ IN DT NNP NNP, IN DT NNP ⟩
16	⟨ JJR, JJ ⟩	⟨ IN DT NN IN, IN NNS IN ⟩
17	⟨ JJR, VBG ⟩	⟨ NNP NNP POS, NNP POS ⟩
18	⟨ CC, VBD ⟩	⟨ NNP NNP IN, NNP IN ⟩
19	⟨ JJR, VBN ⟩	⟨ TO VB DT, TO DT ⟩
20	⟨ DT, JJ ⟩	⟨ IN NN IN, IN NNS IN ⟩
21	⟨ CD, VBG ⟩	⟨ NNS MD, NN MD ⟩
22	⟨ LS, SYM ⟩	⟨ JJ NNS, JJ NN NNS ⟩
23	⟨ NN, JJ ⟩	⟨ JJ NN NN, JJ JJ NN ⟩
24	⟨ VBG, JJ ⟩	⟨ NN NNS, JJ NNS ⟩
25	⟨ JJR, RBR ⟩	⟨ PRP VBZ, PRP VBD ⟩
26	⟨ CC, VBZ ⟩	⟨ NN IN NNP, NN IN NNP NNP ⟩
27	⟨ CC, RB ⟩	⟨ NNP NNP CC, NNP CC ⟩
28	⟨ DT, CD ⟩	⟨ NN VBZ, NN VBD ⟩
29	⟨ NN, NNP ⟩	⟨ IN NNP NNP NNP, IN NNP NNP ⟩
30	⟨ VBG, VBD ⟩	⟨ IN DT JJ NN, IN DT NN NN ⟩
31	⟨ CC, VBG ⟩	⟨ DT JJ NNS, DT NNS ⟩
32	⟨ TO, CC ⟩	⟨ JJ NN, JJ JJ NN ⟩
33	⟨ WRB, VBG ⟩	⟨ DT JJ JJ, PR$ JJ ⟩
34	⟨ CD, NNS ⟩	⟨ VBZ DT, VBD DT ⟩
35	⟨ IN, VBD ⟩	⟨ DT JJ JJ, DT JJ ⟩
36	⟨ RB, NNS ⟩	⟨ CC NNP, CC NNP NNP ⟩
37	⟨ RP, JJR ⟩	⟨ JJ NN, JJ NN NN ⟩
38	⟨ VBZ, VBG ⟩	⟨ DT NNP NN, DT NN NN ⟩
39	⟨ RB, RBR ⟩	⟨ NN IN, NN NN IN ⟩
40	⟨ RP, RBR ⟩	⟨ NN IN DT, NNS IN DT ⟩

Figure 3.2: The most similar part-of-speech pairs and part-of-speech sequence pairs, based on the Jensen-Shannon divergence of their left/right tag signatures.



the corresponding NP node in the tree is followed by a verb phrase. Since we do have gold-standard parse trees for the sentences in the Penn Treebank, we can do the following experiment. For each constituent node x in each treebank parse tree t, we record the yield of x as well as its context, for two definitions of context. First, we look at the local linear context as before. Second, we define the left context of x to be the left sibling of the lowest ancestor of x (possibly x itself) which has a left sibling, or $\diamond$ if x is sentence-initial. We define the right context symmetrically. For example, in the parse tree above, *factory payrolls* is a noun phrase whose lowest right sibling is the VP node, and whose lowest left sibling is the beginning of the sentence. This is the *local hierarchical* context. Figure 3.3 shows the most similar pairs of frequent sequences according to Jensen-Shannon divergence between signatures for these two definitions of context. Since we only took counts for tree nodes x, these lists only contain sequences which are frequently constituents. The lists are relatively similar, suggesting that the patterns detected by the two definitions of context are fairly well correlated, supporting the earlier assumption that the local linear context should be largely sufficient. This correlation is fortunate – some of the methods we will investigate are greatly simplified by the ability to appeal to linear context when hierarchical context might be linguistically more satisfying (see chapter 5).

A final important point is that traditional linguistic argumentation for constituency goes far beyond distributional facts (substitutability). Some arguments, like the tendency of targets of dislocation to be constituents might have distributional correlates. For example, dislocatable sequences might be expected to occur frequently at sentence boundary contexts, or have high context entropy. Other arguments for phrasal categories, like those

Rank	Constituent Sequences by Linear Context	Constituent Sequences by Hierarchical Context
1	⟨ NN NNS, JJ NNS ⟩	⟨ NN NNS, JJ NNS ⟩
2	⟨ IN NN, IN DT NN ⟩	⟨ IN NN, IN DT NN ⟩
3	⟨ DT JJ NN, DT NN ⟩	⟨ IN DT JJ NN, IN JJ NNS ⟩
4	⟨ DT JJ NN, DT NN NN ⟩	⟨ VBZ VBN, VBD VBN ⟩
5	⟨ IN DT JJ NN, IN DT NN ⟩	⟨ NN NNS, JJ NN NNS ⟩
6	⟨ NN NNS, JJ NN NNS ⟩	⟨ DT JJ NN NN, DT NN NN ⟩
7	⟨ IN JJ NNS, IN NNS ⟩	⟨ IN DT JJ NN, IN DT NN ⟩
8	⟨ DT JJ NN NN, DT NN NN ⟩	⟨ IN JJ NNS, IN DT NN ⟩
9	⟨ NNP NNP POS, NNP POS ⟩	⟨ DT JJ NN, DT NN ⟩
10	⟨ IN JJ NNS, IN JJ NN ⟩	⟨ DT JJ NN, DT NN NN ⟩
11	⟨ IN NNP, IN NNP NNP ⟩	⟨ IN NNS, IN NN NNS ⟩
12	⟨ JJ NNS, JJ NN NNS ⟩	⟨ IN NNP, IN NNP NNP ⟩
13	⟨ IN DT JJ NN, IN JJ NNS ⟩	⟨ IN DT NN, IN NNP ⟩
14	⟨ IN NNS, IN NN NNS ⟩	⟨ IN JJ NNS, IN JJ NN ⟩
15	⟨ IN JJ NNS, IN DT NN ⟩	⟨ DT NNP NNP, DT NNP ⟩
16	⟨ DT NNP NNP, DT NNP ⟩	⟨ IN JJ NNS, IN NNS ⟩
17	⟨ JJ NNS, DT NNS ⟩	⟨ IN JJ NNS, IN NNP ⟩
18	⟨ DT JJ NNS, DT NNS ⟩	⟨ VBZ VBN, MD VB ⟩
19	⟨ IN JJ NNS, IN NN ⟩	⟨ JJ NNS, JJ NN NNS ⟩
20	⟨ NN NNS, DT NNS ⟩	⟨ IN DT NN NN, IN DT NN ⟩
21	⟨ IN DT JJ NN, IN NN ⟩	⟨ IN DT NN NN, IN DT NNS ⟩
22	⟨ JJ JJ NNS, JJ NN NNS ⟩	⟨ IN JJ NNS, IN NN ⟩
23	⟨ DT NN POS, NNP NNP POS ⟩	⟨ DT JJ NN, JJ NNS ⟩
24	⟨ IN NNS, IN JJ NN ⟩	⟨ DT NNP NN, DT NN NN ⟩
25	⟨ JJ NN, DT JJ NN ⟩	⟨ JJ NNS, DT NN NN ⟩
26	⟨ IN DT NN NN, IN DT NN ⟩	⟨ DT NNS, DT NN NN ⟩
27	⟨ IN NN NNS, IN JJ NN ⟩	⟨ IN JJ NNS, IN NN NNS ⟩
28	⟨ DT NNP NN, DT NN NN ⟩	⟨ NN NNS, DT NNS ⟩
29	⟨ IN JJ NNS, IN NN NNS ⟩	⟨ IN DT NN NN, IN JJ NN ⟩
30	⟨ IN NN, IN NNS ⟩	⟨ IN DT JJ NN, IN NNP ⟩
31	⟨ IN NN, IN JJ NN ⟩	⟨ IN DT NN NN, IN NN NNS ⟩
32	⟨ JJ NN, DT NN NN ⟩	⟨ DT NNP NNP, DT NNP NN ⟩
33	⟨ VB DT NN, VB NN ⟩	⟨ IN DT NN NN, IN JJ NNS ⟩
34	⟨ IN DT NN NN, IN JJ NN ⟩	⟨ JJ JJ NNS, JJ NN NNS ⟩
35	⟨ DT NN, DT NN NN ⟩	⟨ VBD VBN, VBD JJ ⟩
36	⟨ DT NNP NNP, DT NNP NN ⟩	⟨ IN NN, IN NNP ⟩
37	⟨ JJ JJ NNS, JJ NNS ⟩	⟨ VB DT NN, VB NN ⟩
38	⟨ IN DT JJ NN, IN DT NN NN ⟩	⟨ IN NN NNS, IN JJ NN ⟩
39	⟨ JJ NN, NN NN ⟩	⟨ NN NNS, DT NN NN ⟩
40	⟨ DT JJ NNS, JJ NN NNS ⟩	⟨ IN NN NNS, IN NNP NNP ⟩

Figure 3.3: The most similar sequence pairs, based on the Jensen-Shannon divergence of their signatures, according to both a linear and a hierarchical definition of context.

which reference internal consistency (e.g., noun phrases all having a nominal head), are not captured by distributional similarity, but can potentially be captured in other ways. However, scanning figure 3.2, it is striking that similar pairs do tend to have similar internal structure – the chief difficulty isn't telling that DT JJ NN IN is somehow similar to DT NN IN, it's telling that neither is a constituent.

Chapter 4

A Structure Search Experiment

A broad division in statistical methods for unsupervised grammar induction is between structure search methods and parameter search methods. In structure search, the primary search operator is a symbolic change to the grammar. For example, one might add a production to a context-free grammar. In parameter search, one takes a parameterized model with a fixed topology, and the primary search operator is to nudge the parameters around a continuous space, using some numerical optimization procedure. Most of the time, the optimization procedure is the expectation-maximization algorithm, and it is used to fit a parameterized probabilistic model to the data. A classic instance of this method is estimating the production weights for a PCFG with an *a priori* fixed set of rewrites.

Of course, the division is not perfect – a parameter search can have symbolic effects, for example by zeroing out certain rewrites' probabilities, and a structure search procedure often incorporates parameter search inside each new guess at the symbolic structure. Nonetheless, the distinction is broadly applicable, and the two approaches have contrasting motivations. We will discuss the potential merits of parameter search methods later (section 5.1, section 6.1.2), but their disadvantages are easy to see.

First, the historical/empirical stigma: early attempts at parameter search were extremely discouraging, even when applied to toy problems. Lari and Young (1990) report that, when using EM to recover extremely simple context-free grammars, the learned grammar would require several times the number of non-terminals to recover the structure of a target grammar, and even then it would often learn weakly equivalent variants of that target grammar.

When applied to real natural language data, the results were, unsurprisingly, even worse. Carroll and Charniak (1992) describes experiments running the EM algorithm from random starting points, resulting in widely varying grammars of extremely poor quality (for more on these results, see section 5.1).

Second, parameter search methods all essentially maximize the data likelihood, either conditioned on the model or jointly with the model. Of course, outside of language modeling scenarios, we don't generally care about data likelihood for its own sake – we want our grammars to parse accurately, or be linguistically plausible, or we have some goal extrinsic to the training corpus in front of us. While it's always possible data likelihood in our model family will correspond to whatever our real goal is, in practice it's not guaranteed, and often demonstrably false. As far as it goes, this objection holds equally well for structure search methods which are guided by data- or model-posterior-likelihood metrics. However, in structure search methods one only needs a local heuristic for evaluating symbolic search actions. This heuristic can be anything we want – whether we understand what it's (greedily) maximizing or not. This property invites an approach to grammar induction which is far more readily available in structure search approaches than in parameter search approaches: dream up a local heuristic, grow a grammar using greedy structure search, and hope for the best. To the extent that we can invent a heuristic that embodies our true goals better than data likelihood, we might hope to win out with structure search.[1]

The following chapter is a good faith attempt to engineer just such a structure search system, using the observations in chapter 3. While the system does indeed produce encouragingly linguistically sensible context-free grammars, the structure search procedure turns out to be very fragile and the grammars produced do not successfully cope with the complexities of broad-coverage parsing. Some flaws in our system are solved in various other works; we will compare our system to other structure-search methods in section 4.3. Nonetheless, our experiences with structure search led us to the much more robust parameter search systems presented in later chapters.

[1]Implicit in this argument is the assumption that inventing radical new objectives for parameter search procedures is much harder, which seems to be the case.

4.1 Approach

At the heart of any structure search-based grammar induction system is a method, implicit or explicit, for deciding how to update the grammar. In this section, we attempt to engineer a local heuristic which identifies linguistically sensible grammar changes, then use that heuristic to greedily construct a grammar. The core idea is to use distributional statistics to identify sequences which are likely to be constituents, to create categories (grammar non-terminals) for those sequences, and to merge categories which are distributionally similar.

Two linguistic criteria for constituency in natural language grammars motivate our choices of heuristics (Radford 1988):

1. External distribution: A constituent is a sequence of words which appears in various structural positions (within larger constituents).

2. Substitutability: A constituent is a sequence of words with (simple) variants which can be substituted for that sequence.

To make use of these intuitions, we use a local notion of distributional context, as described in chapter 3. Let α be a part-of-speech tag sequence. Every occurrence of α will be in some context $x\ \alpha\ y$, where x and y are the adjacent tags or sentence boundaries. The distribution over contexts in which α occurs is called its *signature*, which we denote by $\sigma(\alpha)$.

Criterion 1 regards constituency itself. Consider the tag sequences IN DT NN and IN DT. The former is a canonical example of a constituent (of category PP), while the later, though strictly more common, is, in general, not a constituent. Frequency alone does not distinguish these two sequences, but Criterion 1 points to a distributional fact which does. In particular, IN DT NN occurs in many environments. It can follow a verb, begin a sentence, end a sentence, and so on. On the other hand, IN DT is generally followed by some kind of a noun or adjective.

This argument suggests that a sequence's constituency might be roughly indicated by the entropy of its signature, $H(\sigma(\alpha))$. Entropy, however, turns out to be only a weak indicator of true constituency. To illustrate, figure 4.1 shows the actual most frequent constituents

in the WSJ10 data set (see section 2.1.1), along with their rankings by several other measures. Despite the motivating intuition of constituents occurring in many contexts, entropy by itself gives a list that is not substantially better-correlated with the true list than simply listing sequences by frequency. There are two primary causes for this. One is that uncommon but possible contexts have little impact on the tag entropy value, yet in classical linguistic argumentation, configurations which are less common are generally not taken to be less grammatical.

To correct for the empirical skew in observed contexts, let $\sigma_u(\alpha)$ be the uniform distribution over the observed contexts for α. This signature flattens out the information about what contexts are more or less likely, but preserves the count of possible contexts. Using the entropy of $\sigma_u(\alpha)$ instead of the entropy of $\sigma(\alpha)$ would therefore have the direct effect of boosting the contributions of rare contexts, along with the more subtle effect of boosting the rankings of more common sequences, since the available samples of common sequences will tend to have collected nonzero counts of more of their rare contexts. However, while $H(\sigma(\alpha))$ presumably converges to some sensible limit given infinite data, $H(\sigma_u(\alpha))$ will not, as noise eventually makes all or most counts non-zero. Let u be the uniform distribution over all contexts. The scaled entropy

$$H_s(\sigma(\alpha)) = H(\sigma(\alpha))[H(\sigma_u(\alpha))/H(u)]$$

turned out to be a useful quantity in practice.[2] Multiplying entropies is not theoretically meaningful, but this quantity does converge to $H(\sigma(\alpha))$ given infinite (noisy) data. The list for scaled entropy still has notable flaws, mainly relatively low ranks for common NPs, which does not hurt system performance, and overly high ranks for short subject-verb sequences, which does.

The other fundamental problem with these entropy-based rankings stems from the context features themselves. The entropy values will change dramatically if, for example, all noun tags are collapsed, or if functional tags are split. This dependence on the tagset

[2]There are certainly other ways to balance the flattened and unflattened distribution, including interpolation or discounting. We found that other mechanisms were less effective in practice, but little of the following rests crucially on this choice.

Sequence	Actual	Freq	Entropy	Scaled	Boundary
DT NN	1	2	4	2	1
NNP NNP	2	1	-	-	4
CD CD	3	9	-	-	-
JJ NNS	4	7	3	3	2
DT JJ NN	5	-	-	-	10
DT NNS	6	-	-	-	9
JJ NN	7	3	-	7	6
CD NN	8	-	-	-	-
IN NN	9	-	-	9	10
IN DT NN	10	-	-	-	-
NN NNS	-	-	5	6	3
NN NN	-	8	-	10	7
TO VB	-	-	1	1	-
DT JJ	-	6	-	-	-
MD VB	-	-	10	-	-
IN DT	-	4	-	-	-
PRP VBZ	-	-	-	-	8
PRP VBD	-	-	-	-	5
NNS VBP	-	-	2	4	-
NN VBZ	-	10	7	5	-
RB IN	-	-	8	-	-
NN IN	-	5	-	-	-
NNS VBD	-	-	9	8	-
NNS IN	-	-	6	-	-

Figure 4.1: Candidate constituent sequences by various ranking functions. Top non-trivial sequences by actual constituent counts, raw frequency, raw entropy, scaled entropy, and boundary scaled entropy in the WSJ10 corpus. The upper half of the table lists the ten most common constituent sequences, while the bottom half lists all sequences which are in the top ten according to at least one of the rankings.

for constituent identification is very undesirable. One appealing way to remove this dependence is to distinguish only two tags: one for the sentence boundary (#) and another for words. Scaling entropies by the entropy of this reduced signature produces the improved list labeled "Boundary." This quantity was not used in practice because, although it is an excellent indicator of NP, PP, and intransitive S constituents, it gives too strong a bias against other constituents, which do not appear so frequently both sentence-initially and sentence-finally. However, the present system is not driven exclusively by the entropy measure used, and duplicating the above rankings more accurately did not always lead to better end results.

In summary, we have a collection of functions of distributional signatures which loosely, but very imperfectly, seem to indicate the constituency of a sequence.

Criterion 2 suggests we then use similarity of distributional signatures to identify when two constituent sequences are of the same constituent type. This seems reasonable enough – NNP and PRP are both NP yields, and occur in similar environments characteristic of NPs. This criterion has a serious flaw: even if our data were actually generated by a PCFG, it need not be the case that all possible yields of a symbol X will have identical distributions. As a concrete example, PRP and NNP differ in that NNP occurs as a subsequence of longer NPs like NNP NNP, while PRP generally doesn't. The context-freedom of a PCFG process doesn't guarantee that all sequences which are possible NP yields have identical distributions; it only guarantees that the NP occurrences of such sequences have identical distributions. Since we generally don't have this kind of information available in a structure search system, at least to start out with, one generally just has to hope that signature similarity will, in practice, still be reliable as an indicator of syntactic similarity. Figure 3.2 shows that if two sequences have similar raw signatures, then they do tend to have similar syntactic behavior. For example, DT JJ NN and DT NN have extremely similar signatures, and both are common noun phrases. Also, NN IN and NN NN IN have very similar signatures, and both are primarily non-constituents.

Given these ideas, section 4.2 discusses a system, called GREEDY-MERGE, whose grammar induction steps are guided by sequence entropy and interchangeability. The output of GREEDY-MERGE is a symbolic CFG suitable for partial parsing. The rules it learns appear to be of high linguistic quality (meaning they pass the dubious "glance test", see

figures 4.4 and 4.5), but parsing coverage is very low.

4.2 GREEDY-MERGE

GREEDY-MERGE is a precision-oriented system which, to a first approximation, can be seen as an agglomerative clustering process over sequences, where the sequences are taken from increasingly structured analyses of the data. A run of the system shown in figure 4.3 will be used as a concrete example of this process.

We begin with all subsequences occurring in the WSJ10 corpus. For each pair of such sequences, a scaled divergence is calculated as follows:

$$d(\alpha, \beta) = \frac{D_{JS}(\sigma(\alpha),\sigma(\beta))}{H_s(\sigma(\alpha))+H_s(\sigma(\beta))}$$

Small scaled divergence between two sequences indicates some combination of similarity between their signatures and high rank according to the scaled entropy "constituency" heuristic. The pair with the least scaled divergence is selected for merging.[3] In this case, the initial top candidates were

[3]We required that the candidates be among the 250 most frequent sequences. The exact threshold was not important, but without some threshold, long singleton sequences with zero divergence were always chosen. This suggests that we need a greater bias towards quantity of evidence in our basic method.

Rank	Proposed Merge	
1	NN	NN NN
2	NNP	NNP NNP
3	NN	JJ NN
4	NNS	NN NNS
5	NNP NNP	NNP NNP NNP
6	DT NN	DT JJ NN
7	JJ NN	NN NN
8	DT	PRP$
9	DT	DT JJ
10	VBZ	VBD
11	NN	NNS
12	PRP VBD	PRP VBZ
13	VBD	MD VB
14	NNS VBP	NN VBZ
15	DT NN	DT NN NN
16	VBZ	VBZ RB
17	NNP	NNP NNP NNP
18	DT JJ	PRP$
19	IN NN	IN DT NN
20	RB	RB RB

Note that the top few merges are all linguistically sensible noun phrase or $\overline{\text{N}}$ unit merges. Candidates 8, 10, and 11 are reasonable part-of-speech merges, and lower on the list (19) there is good prepositional phrase pair. But the rest of the list shows what could easily go wrong in a system like this one. Candidate 9 suggests a strange determiner-adjective grouping, and many of the remaining candidates either create verb-(ad)verb groups or subject-verb groupings instead of the standard verb-object verb phrases. Either of these kinds of merges will take the learned grammar away from the received linguistic standard. While admittedly neither mistake is really devastating provided the alternate analysis is systematic in the learned grammar, this system has no operators for backing out of early mistakes. At this point, however, only the single pair $\langle \text{NN}, \text{NN NN} \rangle$ is selected.

Merging two sequences involves the creation of a single new non-terminal category for those sequences, which rewrites as either sequence. These created categories have arbitrary names, such as *z17*, though in the following exposition, we give them useful descriptors. In this case, a category *z1* is created, along with the rules

$$\text{z1} \rightarrow \text{NN}$$

$$\text{z1} \rightarrow \text{NN NN}$$

Unary rules are disallowed in this system; a learned unary is instead interpreted as a merge of the parent and child grammar symbols, giving

$$\text{NN} \rightarrow \text{NN NN}$$

This single rule forms the entire grammar after the first merge, and roughly captures how noun-noun compounding works in English: any sequence NN* is legal, and can internally group in any way. The grammar rules are unweighted.

At this point, all the input sentences are re-parsed with the current grammar, using a shallow parser which selects an arbitrary minimum-fragments parse. Most sentences will be almost entirely flat, except for sequences of multiple adjacent nouns, which will be analyzed into chunks by this first rule. Once there are non-terminal categories, and the parses aren't entirely flat, the definitions of sequences and contexts become slightly more complex. The sequences in the model are now contiguous siblings in the current round's parses, including, in general, non-terminal symbols in addition to the original terminal set. The contexts of a sequence can either be the linear context, or the hierarchical context, as defined in section 3.4. To illustrate, in figure 4.2, the sequence VBZ RB can be considered in the local context [NN. . . ◇] or the hierarchical context [z1. . . ◇]. The hierarchical context performed slightly better, and was used for the present experiments. The new sequences and their new signatures were tallied, and another pair was selected for merging.

To fully specify the merging rules, each merge creates a new grammar symbol. Any unaries are treated as collapsing the parent and child symbols. Note that this means that whenever the candidate pair contains a length-one sequence, as in the first merge, the newly created symbol is immediately collapsed and discarded. Furthermore, merging two length-one sequences collapses the two symbols in the grammar, so actually reduces the non-terminal vocabulary by one. In the present example, this situation happens for the first time on step 4, where the verbal tags VBZ and VBD are merged. After a merge, re-analysis of the right hand sides of the grammar rules is in general necessary. Any rule which can be parsed by the other rules of the grammar is parsed and simplified. For example, in step 14, common noun phrases with and without determiners are identified, triggering a re-parsing

of the ZNP → DT JJ NN rule into ZNP → DT ZNP.[4]

Eyeballing the merges chosen, the initial choices of this procedure look plausible. Noun chunks are identified (1,2), then determiner-bearing noun phrases (3), then some tag distinctions which encode feature and tense are collapsed (4,5). Prepositional phrases are identified in (7), verb-object verb phrases in (10), and NP/VP sentence structures in (20). Some (relatively minor) missteps along the way include a non-standard verb group chunk in (6) and a (worse) determiner-adjective chunk in (12). Then there is a combination of these categories being fleshed out (usually sensibly) and merged together (usually overly aggressively). Starting on merge (45), the system begins imploding, merging nouns with noun phrases, then adverbs, and so on, until merge (54), where nouns (along with most other tags) are merged with verbs. At this point, all the top matches are longer, as-yet-unanalyzed sequences, but the initially promising grammar has mostly collapsed into itself. This behavior underscores that for the GREEDY-MERGE system, stopping at the correct point is critical. Since our greedy criterion is not a measure over entire grammar states, we have no way to detect the optimal point beyond heuristics (the same category appears in several merges in a row, for example) or by using a small supervision set to detect a parse performance drop.

In addition to eyeballing the merges and grammars at each stage, a more quantifiable way to monitor the coverage and accuracy of our grammar as it grows is to take the output of the partial parser, and compare those (initially shallow) trees to the gold standard. Figure 4.3 shows unlabeled precision and recall after each stage. These figures ignore the labels on the proposed trees, and ignore all brackets of size one (but not full-sentence brackets, which all partial parses have, which gives the non-zero initial recall). Overall, as the grammar grows, we trade the initially perfect precision for recall, substantially increasing F_1 until step (15). Then, the trade-off continues, with F_1 more constant until about step (27), at which point F_1 begins to decline. By step (54) where nouns et al. are merged with verbs et al., F_1 has dropped from a high of 56.5 down to a low of 33.7.

[4]This is a case where we really would rather preserve a unary that represents a null determiner.

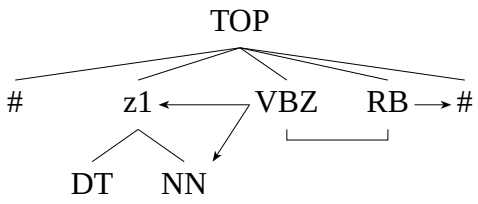


Figure 4.2: Two possible contexts of a sequence: linear and hierarchical.

4.2.1 Grammars learned by GREEDY-MERGE

Figure 4.4 shows a snapshot of the grammar at one stage of a run of GREEDY-MERGE on the WSJ10 corpus.[5] The non-terminal categories proposed by the systems are internally given arbitrary designations, but we have relabeled them to indicate what standard classes they best correspond to.

Categories corresponding to NP, VP, PP, and S are learned, although some are split into sub-categories (transitive and intransitive VPs, proper NPs and two kinds of common NPs, and so on).[6] NPs have internal structure where adnominal modifiers are grouped with nouns, determiners attached higher, and verbs are chunked into verb groups (contrary to most but not all traditional linguistic argumentation, (Halliday 1994, Radford 1988)). Provided one is willing to accept such a verb-group analysis, this grammar seems sensible, though quite a few constructions, such as relative clauses, are missing entirely.

Figure 4.5 shows a grammar learned at one stage of a run when verbs were split by transitivity. This grammar is similar, but includes analyses of sentential coordination and adverbials, and subordinate clauses. The only rule in this grammar which seems overly suspect is ZVP → IN ZS which analyzes complementized subordinate clauses as VPs.

In general, the major mistakes the GREEDY-MERGE system makes are of three sorts:

- Mistakes of omission. Even though the grammar snapshots shown have correct, recursive analyses of many categories, neither has rules which can non-trivially incorporate a number (CD). There is also no analysis for many common constructions, including relative clauses, comparatives, and, most worryingly, conjunctions.

[5] This grammar, while very similar, does not exactly match the full run shown in figure 4.3, but reflects slightly different parameter settings.

[6] Splits often occur because unary rewrites are not learned in this system.

Step	Merged Sequences		Resulting Rules	UPrec.	URec.	F_1
0	(original)		(none)	100.0	20.5	34.1
1	NN	NN NN	NN → NN NN	92.3	21.0	34.2
2	NNP	NNP NNP	NNP → NNP NNP	83.8	23.8	37.1
3	DT NN	DT JJ NN	zNP → DT NN, zNP → DT JJ NN	85.6	31.2	45.7
4	VBZ	VBD	(merge)	85.6	31.2	45.7
5	NNS	NN	(merge)	83.8	33.7	48.1
6	VBZ	MD VB	VBZ → MD VB	81.4	33.8	47.8
7	IN NNS	IN zNP	zPP → IN NNS, zPP → IN zNP	81.1	37.0	50.8
8	DT	PRP$	(merge)	81.5	38.2	52.1
9	VBZ	VBP	(merge)	81.6	38.3	52.1
10	VBZ NNS	VBZ zNP	zVP → VBZ NNS, zVP → VBZ zNP	78.4	39.7	52.7
11	VBZ	VBZ RB	VBZ → VBZ RB	74.1	40.1	52.0
12	DT	DT JJ	DT → DT JJ	72.7	40.4	52.0
13	DT	NNP POS	DT → NNP POS	71.0	41.6	52.5
14	zNP	JJ NNS	zNP → JJ NNS	71.0	45.2	55.2
15	zVP	VBZ zPP	zVP → VBZ zPP	70.8	45.6	55.5
16	VBZ	VBZ VBN	VBZ → VBZ VBN	69.0	46.8	55.8
17	zVP	VBZ JJ	zVP → VBZ JJ	68.7	47.9	56.5
18	VBZ	VBZ VBG	VBZ → VBZ VBG	67.7	48.5	56.5
19	zVP	RB zVP	zVP → RB zVP	67.1	48.5	56.4
20	PRP zVP	zNP zVP	zS → PRP zVP, zS → zNP zVP	64.8	48.9	55.7
21	zS	NNP zVP	zS → NNP zVP	64.0	48.9	55.5
22	zS	DT zVP	zS → DT zVP	63.8	48.9	55.4
23	VBZ	VBZ TO VB	VBZ → VBZ TO VB	63.1	49.4	55.4
24	zS	RB zS	zS → RB zS	63.0	49.5	55.4
25	zPP	IN NNP	zPP → IN NNP	63.1	50.6	56.2
26	zNP	DT NNP NNS	zNP → DT NNP NNS	63.1	51.1	56.5
27	zS	NNS zVP	zS → NNS zVP	62.4	51.2	56.3
28	zNP VBZ	NNP VBZ	zSi → zNP VBZ, zSi → NNP VBZ	60.1	51.2	55.3
29	PRP VBZ	zSi	zSi → PRP VBZ	58.9	51.5	54.9
30	zSi	RB zSi	zSi → RB zSi	58.8	51.5	54.9
31	zS	zS zPP	zS → zS zPP	58.3	51.5	54.7
32	VBZ	MD RB VB	VBZ → MD RB VB	58.0	51.8	54.7
33	VBG	VBN	(merge)	58.0	51.8	54.7
34	VBG	TO VB	VBG → TO VB	57.8	52.0	54.7
35	VBZ	zVP	(merge)	53.6	50.9	52.2
36	zS	zSi	(merge)	53.3	50.6	51.9
37	RB	VBG	(merge)	53.2	50.7	51.9
38	zS VBZ	zS zS	zX → zS VBZ, zX → zS zS	52.8	50.8	51.8
39	zS	zX	(merge)	52.8	50.8	51.8
40	MD	MD RB	MD → MD RB	52.6	50.8	51.7
41	zS	DT zS	zS → DT zS	51.8	50.3	51.1
42	zS	zPP zS	zS → zPP zS	51.6	50.3	50.9
43	zS	NNP zS	zS → NNP zS	51.0	50.0	50.5
44	NNP	NNPS	(merge)	50.9	50.2	50.6
45	zS	CC zS	zS → CC zS	50.4	50.2	50.3
46	NNS	zNP	(merge)	50.4	50.6	50.5
47	NNS	RB	(merge)	47.8	49.7	48.8
48	NNS	JJ	(merge)	46.5	48.8	47.6
49	NNS	zPP	(merge)	41.8	44.7	43.2
50	NNS	JJR	(merge)	41.9	45.1	43.5
51	NNS	DT	(merge)	36.3	39.4	37.8
52	NNS	IN	(merge)	33.9	37.5	35.6
53	NNS	JJS	(merge)	33.5	37.3	35.3
54	VBZ	zS	(merge)	34.1	38.0	35.9
55	NNS	VBZ	(merge)	31.7	35.9	33.7
56	RBR TO	LS NNS LS	z111 → RBR TO, z111 → LS NNS LS	31.8	36.1	33.8
57	VB VB VB	WRB NNS RP	z113 → VB VB VB, z113 → WRB NNS RP	31.9	36.2	33.9
58	SYM NNS CD CD	NNS WP NNS CD	z115 → SYM NNS CD CD, z115 → NNS WP NNS CD	32.3	36.7	34.4
59	WRB NNS PRP VB	NNS PDT NNS TO	z117 → WRB NNS PRP VB, z117 → NNS PDT NNS TO	32.5	36.9	34.5
60	117	WRB NNS TO	z117 → WRB NNS TO	32.6	37.0	34.7
61	NNS 113	NNS PRP VB PRP	z121 → NNS z113, z121 → NNS PRP VB PRP	32.7	37.1	34.7
62	z115	NNS VB NNS RBR	z115 → NNS VB NNS RBR	32.9	37.4	35.0
63	z115	NNS VB NNS WRB	z115 → NNS VB NNS WRB	32.6	37.0	34.7
64	z117	NNS UH TO	z117 → NNS UH TO	33.0	37.5	35.1
65	z115	NNS VB NNS VB	z115 → NNS VB NNS VB	32.8	37.2	34.9

Figure 4.3: A run of the GREEDY-MERGE system.

N-bar or zero determiner NP
zNN → NN | NNS
zNN → JJ zNN
zNN → zNN zNN

NP with determiner
zNP → DT zNN
zNP → PRP$ zNN

Proper NP
zNNP → NNP | NNPS
zNNP → zNNP zNNP

PP
zPP → zIN zNN
zPP → zIN zNP
zPP → zIN zNNP

verb groups / intransitive VPs
zV → VBZ | VBD | VBP
zV → MD VB
zV → MD RB VB
zV → zV zRB
zV → zV zVBG

Transitive VPs
(complementation)
zVP → zV JJ
zVP → zV zNP
zVP → zV zNN
zVP → zV zPP

Transitive VPs
(adjunction)
zVP → zRB zVP
ZVP → zVP zPP

Intransitive S
zSi → PRP zV
zSi → zNP zV
zSi → zNNP zV

Transitive S
zS → zNNP zVP
zS → zNN zVP
zS → PRP zVP

Figure 4.4: A grammar learned by GREEDY-MERGE.

N-bar or zero-determiner NP
zNN → NN | NNS
zNN → zNN zNN
zNN → JJ zNN

Common NP with determiner
zNP → DT zNN
zNP → PRP$ zNN

Proper NP
zNNP → zNNP zNNP
zNNP → NNP

PP
zPP → zIN zNN
zPP → zIN zNP
zPP → zIN zNNP

Transitive Verb Group
zVt → VBZt | VBDt | VBPt
zVt → MD zVBt
zVt → zVt RB

Intransitive Verb Group
zVP → VBZ | VBD | VBP
zVP → MD VB
zVP → zVP zVBN [1]

VP adjunction
zVP → RB zVP
zVP → zVP RB
zVP → zVP zPP
zVP → zVP zJJ

VP complementation
zVP → zVt zNP
zVP → zVt zNN

S
zS → zNNP zVP
zS → zNN zVP
zS → zNP zVP
zS → DT zVP

zS → CC zS
zS → RB zS

S-bar
zVP → IN zS [2]

1 - wrong attachment level
2 - wrong result category

Figure 4.5: A grammar learned by GREEDY MERGE (with verbs split by transitivity).

- Alternate analyses. The system almost invariably forms verb groups, merging MD VB sequences with single main verbs to form verb group constituents (argued for at times by some linguists (Halliday 1994)). Also, PPs are sometimes attached to NPs below determiners (which is in fact a standard linguistic analysis (Abney 1987)). It is not always clear whether these analyses should be considered mistakes.

- Over-merging. These errors are the most serious. Since at every step two sequences are merged, the process will eventually learn the grammar where X → X X and X → (any terminal). However, very incorrect merges are sometimes made relatively early on (such as merging VPs with PPs, or merging the sequences IN NNP IN and IN).

A serious issue with GREEDY-MERGE is that the grammar learned is symbolic, not probabilistic. Any disambiguation is done arbitrarily. Therefore, even adding a linguistically valid rule can degrade numerical performance (sometimes dramatically) by introducing ambiguity to a greater degree than it improves coverage. This issue, coupled with the many omissions in these grammars, emphasizes the degree to which eyeballing grammar snapshots can be misleadingly encouraging.

4.3 Discussion and Related Work

There is a great deal of previous work on structure-search methods, and it must be emphasized that while the preceding system is broadly representative, many of its flaws are overcome by some prior work or other. Wolff (1988) presents an overview of much of Wolff's work up to that point. His program SNPR is a chunking system, which has two operations: folding, which is like the merge above, and generalization, which is like the reparsing step above. His system is not statistical, though it does prioritize operations based on corpus frequency. The most striking idea present in his work which is missing here is that generalizations which are not fully attested can be retracted in a repair operation, allowing, in principle, for early mistakes to be undone later in the process. His work is intended to be a cognitively plausible account of language acquisition using minimal native bias. Other authors guide structure searches using an explicit compression-based criterion,

preferring to introduce rules which increase the likelihood of the grammar given the data. (Stolcke and Omohundro 1994) describes Bayesian model-merging, where the increase in data likelihood is balanced against an MDL-style prior over models (which prefers simpler models). Chen (1995), Kit (1998), and Langley and Stromsten (2000) present more recent MDL approaches; these methods have in common that they do not seem to scale to real text, and can suffer from the tendency to chunk common functional units, like IN DT, together early on. As Alex Clark has pointed out (Clark 2001b), it is not the use of MDL that is problematic, but rather its *greedy* use. Magerman and Marcus (1990), which is otherwise along the same lines, has an innovative mal-rule approach which forbids certain such problematic sequences from being wrongly analyzed as constituents. Finally, Clark (2001a) presents a hybrid system which uses an MDL search in conjunction with distributional methods (see chapter 3). For a more thorough survey, see Clark (2001b).

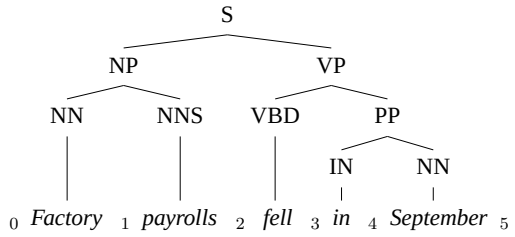
Chapter 5

Constituent-Context Models

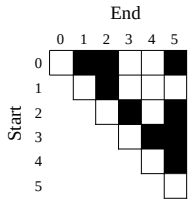
5.1 Previous Work

In contrast with the relative success of word-class learning methods, induction of syntax has proven to be extremely challenging. Some of the earliest and most important signs of discouragement from statistical parameter search methods were the results of Lari and Young (1990). Their work showed that even simple artificial grammars could not be reliably recovered using EM over the space of PCFGs (using the inside-outside algorithm: see Manning and Schütze (1999) for an introduction). The problem wasn't with the model family: Charniak (1996) showed that a maximum-likelihood PCFG read off of a treebank could parse reasonably well, and most high-performance parsers have, strictly speaking, been in the class of PCFG parsing. Therefore the problem was either with the use of EM as a search procedure or with some mismatch between data likelihood and grammar quality. Either way, their work showed that simple grammars were hard to recover in a fully unsupervised manner.

Carroll and Charniak (1992) tried the PCFG induction approach on natural language data, again with discouraging results. They used a structurally restricted PCFG in which the terminal symbols were parts-of-speech and the non-terminals were part-of-speech projections. That is, for every part-of-speech X there was a non-terminal $\overline{\text{X}}$, with the rewrites restricted to the forms $\overline{\text{X}} \rightarrow \overline{\text{X}}\ \overline{\text{Y}}$ and $\overline{\text{X}} \rightarrow \overline{\text{Y}}\ \overline{\text{X}}$ (plus unary terminal rewrites of the form $\overline{\text{X}} \rightarrow$ X). These configurations can be thought of as head-argument attachments, where X is



(a)



(b)

Span	Label	Constituent	Context
⟨0,5⟩	S	NN NNS VBD IN NN	◇ – ◇
⟨0,2⟩	NP	NN NNS	◇ – VBD
⟨2,5⟩	VP	VBD IN NN	NNS – ◇
⟨3,5⟩	PP	IN NN	VBD – ◇
⟨0,1⟩	NN	NN	◇ – NNS
⟨1,2⟩	NNS	NNS	NN – VBD
⟨2,3⟩	VBD	VBD	NNS – IN
⟨3,4⟩	IN	IN	VBD – NN
⟨4,5⟩	NN	NNS	IN – ◇

(c)

Figure 5.1: Parses, bracketings, and the constituent-context representation for the sentence, "Factory payrolls fell in September." Shown are (a) an example parse tree, (b) its associated bracketing, and (c) the yields and contexts for each constituent span in that bracketing. Distituent yields and contexts are not shown, but *are* modeled.

the head. In fact, trees in this grammar are isomorphic to dependency trees which specify the attachment order for heads with multiple arguments (Miller 1999). The hope was that, while the symbols in an arbitrary PCFG do not have any *a priori* meaning or structural role, symbols in this dependency grammar are not structurally symmetric – each one is anchored to a specific terminal symbol. Carroll and Charniak describe experiments where many such grammars were weighted randomly, then re-estimated using EM. The resulting grammars exhibited wide variance in the structures learned and in the data likelihood found. Parsing performance was consistently poor (according to their qualitative evaluation). Their conclusion was that the blame lay with the structure search problem: EM is a local maximization procedure, and each initial PCFG converged to a different final grammar. Regardless of the cause, the results did not suggest that PCFG induction was going to be straightforwardly effective.

Other related parameter search work is discussed in section 6.1.2, but it is worth further considering the Carroll and Charniak experiments and results here. One important advantage of their formulation (that they did not exploit) is that random initial rule weights are not actually needed. In the case of unrestricted binary-branching PCFGs, such as with the Lari and Young (1990) experiments, one considers a full binary grammar over symbols $\{\mathrm{X}_i\}$. If all rules $\mathrm{X}_i \rightarrow \mathrm{X}_j\ \mathrm{X}_k$ have exactly equal initial probability, that initial parameter vector will be an unstable symmetric fixed point for EM. Therefore random noise is required for symmetry-breaking, to get the optimization started. That is not the case for the Carroll and

Charniak grammars. While the parameter space is certainly riddled with local maxima, and therefore the initial grammar weights do matter, there is an obvious uniform starting point, where all rules have equal starting probability. Beginning from that uniform initializer, EM will find some solution which we might hope will correspond to a higher quality grammar than most random initializations produce. This hope is borne out in practice: as figure 5.4 shows under the name DEP-PCFG, their method substantially outperforms a random baseline. It does not break the right-branching baseline, however, and we can ask why that might be. One cause is certainly that the grammar itself is representationally lacking; we will discuss this further in chapter 6. Section 5.3.6 discusses another possible issue: a flat grammar initialization gives rise to a very un-language-like posterior over trees.

The distributional clustering of words (chapter 3) has proven remarkably robust for discovering patterns which at least broadly approximate classical parts-of-speech. It is therefore very appealing to try to extend linear distributional techniques to levels of syntax higher than word-classes. Recall the left column of figure 3.3, which shows the most similar tag sequences according to the Jensen-Shannon divergence of their local linear tag signatures. This list makes one optimistic that constituent sequences with very similar contexts will tend to be of the same constituent type. For example, the top three pairs are noun groups, prepositional phrases, and determiner-carrying noun phrases. The subsequent examples include more correct noun and prepositional phrase pairs, with some possessive constructions and verb phrases scattered among them. Indeed, the task of taking constituent sequences and clustering them into groups like noun-phrases and verb phrases is not much harder than clustering words into word classes. The problem is that to produce lists like these, we need to know which subspans of each sentence are constituents. If we simply consider all subspans of the sentences in our corpus, most sequence tokens will not be constituent tokens. The right column of figure 3.2 shows the sequence pairs with most similar contexts, using all subspans instead of constituent subspans. Again we see pairs of similar constituents, like the first pair of proper noun phrases. However, we also see examples like the second pair, which are two non-constituent sequences. It's no surprise these non-constituent, or *distituent*, pairs have similar context distributions – if we had to classify them, they are in some sense similar. But a successful grammar induction system must somehow learn which sequence types should be regularly used in building trees, and which

should not. That is, we need to form coherent tree-structured analyses, and distributional clustering of sequences, robust though it may be, will not give us trees.

One way to get around this limitation of distributional clustering is to first group sequences into types by signature similarity, then differentiate the "good" and "bad" constituent types by some other mechanism. A relatively successful approach along these lines is described in Clark (2001a). Clark first groups sequence types, then uses a mutual information criterion to filter constituents from distituents. The good sequences are then used to build up a PCFG according to a MDL measure. The experimental results of Clark's system are discussed later in this chapter, but the overall parsing performance is rather low because the discovered grammars are extremely sparse.

5.2 A Generative Constituent-Context Model

In this chapter, we describe a model which is designed to combine the robustness of distributional clustering with the coherence guarantees of parameter search. It is specifically intended to produce a more felicitous search space by removing as much hidden structure as possible from the syntactic analyses. The fundamental assumption is a much weakened version of a classic linguistic constituency tests (Radford 1988): constituents appear in constituent context. A particular linguistic phenomenon that the system exploits is that long constituents often have short, common equivalents, or *proforms*, which appear in similar contexts and whose constituency is easily discovered (or guaranteed). Our model is designed to transfer the constituency of a sequence directly to its containing context, which is intended to then pressure new sequences that occur in that context into being parsed as constituents in the next round. The model is also designed to exploit the successes of distributional clustering, and can equally well be viewed as doing distributional clustering in the presence of no-overlap constraints.

5.2.1 Constituents and Contexts

Unlike a PCFG, our model describes all contiguous subsequences of a sentence (*spans*), including empty spans, whether they are constituents or distituents. A span encloses a sequence of terminals, or *yield*, α, such as DT JJ NN. A span occurs in a *context* x, such as $\diamond$–VBZ, where x is the ordered pair of preceding and following terminals ($\diamond$ denotes a sentence boundary). A *bracketing* of a sentence is a boolean matrix B, which indicates which spans are constituents and which are not. Figure 5.1 shows a parse of a short sentence, the bracketing corresponding to that parse, and the labels, yields, and contexts of its constituent spans.

Figure 5.2 shows several bracketings of the sentence in figure 5.1. A bracketing B of a sentence is *non-crossing* if, whenever two spans cross, at most one is a constituent in B. A non-crossing bracketing is *tree-equivalent* if the size-one terminal spans and the full-sentence span are constituents, and all size-zero spans are distituents. Figure 5.2(a) and (b) are tree-equivalent. Tree-equivalent bracketings B correspond to (unlabeled) trees in the obvious way. A bracketing is *binary* if it corresponds to a binary tree. Figure 5.2(b) is binary. We will induce trees by inducing tree-equivalent bracketings.

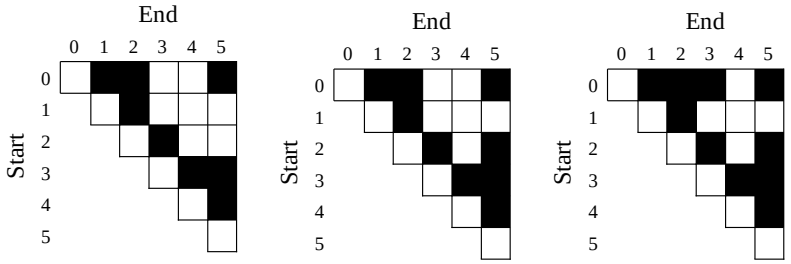
Our generative model over sentences S has two phases. First, we choose a bracketing B according to some distribution $\mathrm{P}(B)$ and then generate the sentence given that bracketing:

$$\mathrm{P}(S, B) = \mathrm{P}(B)\mathrm{P}(S|B)$$

Given B, we fill in each span independently. The context and yield of each span are independent of each other, and generated conditionally on the constituency B_{ij} of that span.

$$\begin{aligned} \mathrm{P}(S|B) &= \prod_{\langle i,j\rangle \in spans(S)} \mathrm{P}(\alpha_{ij}, x_{ij}|B_{ij}) \\ &= \prod_{\langle i,j\rangle} \mathrm{P}(\alpha_{ij}|B_{ij})\mathrm{P}(x_{ij}|B_{ij}) \end{aligned}$$

The distribution $\mathrm{P}(\alpha_{ij}|B_{ij})$ is a pair of multinomial distributions over the set of all possible yields: one for constituents ($B_{ij} = c$) and one for distituents ($B_{ij} = d$). Similarly for $\mathrm{P}(x_{ij}|B_{ij})$ and contexts. The marginal probability assigned to the sentence S is given by summing over all possible bracketings of S: $\mathrm{P}(S) = \sum_B \mathrm{P}(B)\mathrm{P}(S|B)$. Note that this is



(a) Tree-equivalent (b) Binary (c) Crossing

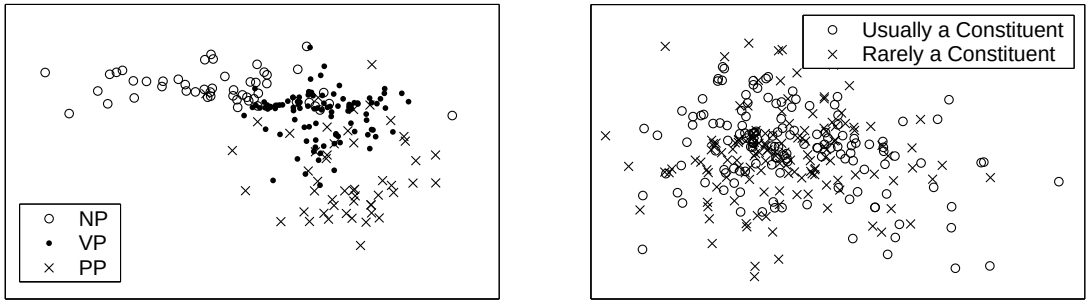
Figure 5.2: Three bracketings of the sentence "Factory payrolls fell in September." Constituent spans are shown in black. The bracketing in (b) corresponds to the binary parse in figure 5.1; (a) does not contain the $\langle 2,5 \rangle$ VP bracket, while (c) contains a $\langle 0,3 \rangle$ bracket crossing that VP bracket.

a more severe set of independence assumptions than, say, in a naive-bayes model. There, documents positions are filled independently, and the result can easily be an ungrammatical document. Here, the result need not even be a structurally consistent sentence.[1]

To induce structure, we run EM over this model, treating the sentences S as observed and the bracketings B as unobserved. The parameters Θ of the model are the constituency-conditional yield and context distributions $\mathrm{P}(\alpha|b)$ and $\mathrm{P}(x|b)$. If $\mathrm{P}(B)$ is uniform over all (possibly crossing) bracketings, then this procedure will be equivalent to soft-clustering with two equal-prior classes.

There is reason to believe that such soft clusterings alone will not produce valuable distinctions, even with a significantly larger number of classes. The distituents must necessarily outnumber the constituents, and so such distributional clustering will result in mostly distituent classes. Clark (2001a) finds exactly this effect, and must resort to a filtering heuristic to separate constituent and distituent clusters. To underscore the difference between the bracketing and labeling tasks, consider figure 5.3. In both plots, each point is a frequent tag sequence, assigned to the (normalized) vector of its context frequencies. Each plot has been projected onto the first two principal components of its respective data set. The left

[1]Viewed as a model generating *sentences*, this model is deficient, placing mass on yield and context choices which will not tile into a valid sentence, either because specifications for positions conflict or because yields of incorrect lengths are chosen. We might in principle renormalize by dividing by the mass placed on proper sentences and zeroing the probability of improper bracketings. In practice, there does not seem to be an easy way to carry out this computation.



(a) Constituent Types (b) Constituents vs. Distituents

Figure 5.3: Clustering vs. detecting constituents. The most frequent yields of (a) three constituent types and (b) constituents and distituents, as context vectors, projected onto their first two principal components. Clustering is effective at labeling, but not detecting, constituents.

plot shows the most frequent sequences of three constituent types. Even in just two dimensions, the clusters seem coherent, and it is easy to believe that they would be found by a clustering algorithm in the full space. On the right, sequences have been labeled according to whether their occurrences are constituents more or less of the time than a cutoff (of 0.2). The distinction between constituent and distituent seems much less easily discernible.

We can turn what at first seems to be distributional clustering into tree induction by confining $\mathrm{P}(B)$ to put mass only on tree-equivalent bracketings. In particular, consider $\mathrm{P}_{\mathrm{bin}}(B)$ which is uniform over binary bracketings and zero elsewhere. If we take this bracketing distribution, then when we sum over data completions, we will only involve bracketings which correspond to valid binary trees. This restriction is the basis for the next algorithm.

5.2.2 The Induction Algorithm

We now essentially have our induction algorithm. We take $\mathrm{P}(B)$ to be $\mathrm{P}_{\mathrm{bin}}(B)$, so that all binary trees are equally likely. We then apply the EM algorithm:

E-Step: Find the conditional completion likelihoods $\mathrm{P}(B|S,\Theta)$ according to the current Θ.

M-Step: Fix $\mathrm{P}(B|S,\Theta)$ and find the Θ' which maximizes $\sum_B \mathrm{P}(B|S,\Theta)\log \mathrm{P}(S,B|\Theta')$.

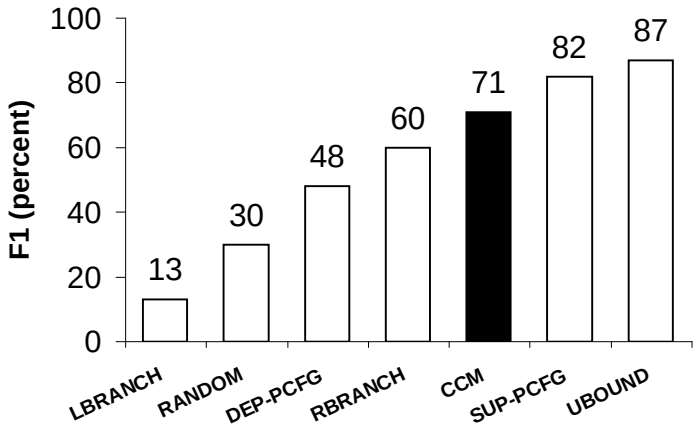


Figure 5.4: Bracketing F_1 for various models on the WSJ10 data set.

The completions (bracketings) cannot be efficiently enumerated, and so a cubic dynamic program similar to the inside-outside algorithm is used to calculate the expected counts of each yield and context, both as constituents and distituents (see the details in appendix A.1). Relative frequency estimates (which are the ML estimates for this model) are used to set Θ'.

5.3 Experiments

The experiments that follow used the WSJ10 data set, as described in chapter 2, using the alternate unlabeled metrics described in section 2.2.5, with the exception of figure 5.15 which uses the standard metrics, and figure 5.6 which reports numbers given by the EVALB program. The basic experiments do not label constituents. An advantage to having only a single constituent class is that it encourages constituents of one type to be proposed even when they occur in a context which canonically holds another type. For example, NPs and PPs both occur between a verb and the end of the sentence, and they can transfer constituency to each other through that context.

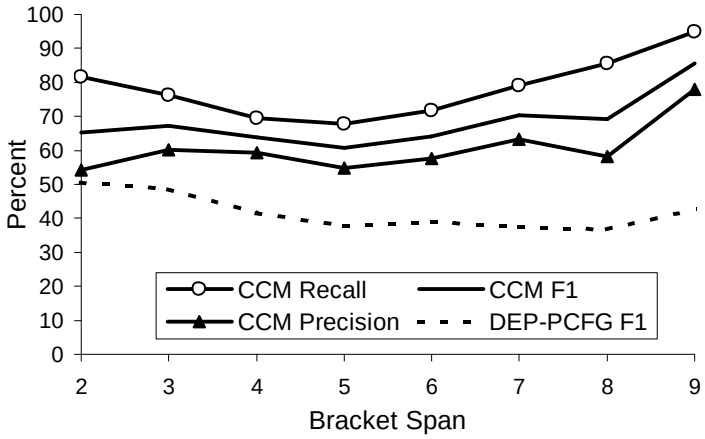


Figure 5.5: Scores for CCM-induced structures by span size. The drop in precision for span length 2 is largely due to analysis inside NPs which is omitted by the treebank. Also shown is F_1 for the induced PCFG. The PCFG shows higher accuracy on small spans, while the CCM is more even.

Figure 5.4 shows the F_1 score for various methods of parsing. RANDOM chooses a tree uniformly at random from the set of binary trees.[2] This is the unsupervised baseline. DEP-PCFG is the result of duplicating the experiments of Carroll and Charniak (1992), using EM to train a dependency-structured PCFG. LBRANCH and RBRANCH choose the left- and right-branching structures, respectively. RBRANCH is a frequently used baseline for *supervised* parsing, but it should be stressed that it encodes a significant fact about English structure, and an induction system need not beat it to claim a degree of success. CCM is our system, as described above. SUP-PCFG is a supervised PCFG parser trained on a 90-10 split of this data, using the treebank grammar, with the Viterbi parse right-binarized.[3] UBOUND is the upper bound of how well a binary system can do against the treebank sentences, which are generally flatter than binary, limiting the maximum precision.

CCM is doing quite well at 71.1%, substantially better than right-branching structure. One common issue with grammar induction systems is a tendency to chunk in a bottom-up fashion. Especially since the CCM does not model recursive structure explicitly, one might be concerned that the high overall accuracy is due to a high accuracy on short-span

[2]This is different from making random parsing decisions, which gave a higher score of 35%.

[3]Without post-binarization, the F_1 score was 88.9.

System	UP	UR	F_1	CB
EMILE	51.6	16.8	25.4	**0.84**
ABL	43.6	35.6	39.2	2.12
CDC-40	53.4	34.6	42.0	1.46
RBRANCH	39.9	46.4	42.9	2.18
CCM	**55.4**	**47.6**	**51.2**	1.45

Figure 5.6: Comparative ATIS parsing results.

constituents. Figure 5.5 shows that this is not true. Recall drops slightly for mid-size constituents, but longer constituents are as reliably proposed as short ones. Another effect illustrated in this graph is that, for span 2, constituents have low precision for their recall. This contrast is primarily due to the single largest difference between the system's induced structures and those in the treebank: the treebank does not parse into NPs such as DT JJ NN, while our system does, and generally does so correctly, identifying $\overline{\text{N}}$ units like JJ NN. This overproposal drops span-2 precision. In contrast, figure 5.5 also shows the F_1 for DEP-PCFG, which does exhibit a drop in F_1 over larger spans.

The top row of figure 5.8 shows the recall of non-trivial brackets, split according the brackets' labels in the treebank. Unsurprisingly, NP recall is highest, but other categories are also high. Because we ignore trivial constituents, the comparatively low S represents *only* embedded sentences, which are somewhat harder even for supervised systems.

To facilitate comparison to other recent work, figure 5.6 shows the accuracy of our system when trained on the same WSJ data, but tested on the ATIS corpus, and evaluated according to the EVALB program. EMILE and ABL are lexical systems described in (van Zaanen 2000, Adriaans and Haas 1999), both of which operate on minimal pairs of sentences, deducing constituents from regions of variation. CDC-40, from (Clark 2001a), reflects training on much more data (12M words), and is describe in section 5.1. The F_1 numbers are lower for this corpus and evaluation method.[4] Still, CCM beats not only RBRANCH (by 8.3%), but the next closest unsupervised system by slightly more.

[4]The primary cause of the lower F_1 is that the ATIS corpus is replete with span-one NPs; adding an extra bracket around *all* single words raises our EVALB recall to 71.9; removing all unaries from the ATIS gold standard gives an F_1 of 63.3%.

Rank	Overproposed	Underproposed
1	JJ NN	NNP POS
2	MD VB	TO CD CD
3	DT NN	NN NNS
4	NNP NNP	NN NN
5	RB VB	TO VB
6	JJ NNS	IN CD
7	NNP NN	NNP NNP POS
8	RB VBN	DT NN POS
9	IN NN	RB CD
10	POS NN	IN DT

Figure 5.7: Constituents most frequently over- and under-proposed by our system.

5.3.1 Error Analysis

Parsing figures can only be a component of evaluating an unsupervised induction system. Low scores may indicate systematic alternate analyses rather than true confusion, and the Penn treebank is a sometimes arbitrary or even inconsistent gold standard. To give a better sense of the kinds of errors the system is or is not making, we can look at which sequences are most often overproposed, or most often underproposed, compared to the treebank parses.

Figure 5.7 shows the 10 most frequently over- and under-proposed sequences. The system's main error trends can be seen directly from these two lists. It forms MD VB verb groups systematically, and it attaches the possessive particle to the right, like a determiner, rather than to the left.[5] It provides binary-branching analyses within NPs, normally resulting in correct extra $\overline{\text{N}}$ constituents, like JJ NN, which are not bracketed in the treebank. More seriously, it tends to attach post-verbal prepositions to the verb and gets confused by long sequences of nouns. A significant improvement over some earlier systems (both ours and other researchers') is the absence of subject-verb groups, which disappeared when we switched to $\mathrm{P}_{\mathrm{split}}(B)$ for initial completions (see section 5.3.6); the more balanced subject-verb analysis had a substantial combinatorial advantage with $\mathrm{P}_{\mathrm{bin}}(B)$.

[5] Linguists have at times argued for both analyses: Halliday (1994) and Abney (1987), respectively.

5.3.2 Multiple Constituent Classes

We also ran the system with multiple constituent classes, using a slightly more complex generative model in which the bracketing generates a labeling L (a mapping from spans to label classes C) which then generates the constituents and contexts.

$$\mathrm{P}(S, L, B) = \mathrm{P}(B)\mathrm{P}(L|B)\mathrm{P}(S|L)$$

$$\mathrm{P}(L|B) = \prod_{\langle i,j\rangle \in spans(S)} \mathrm{P}(L_{ij}|B_{ij})$$

$$\begin{aligned}\mathrm{P}(S|L) &= \prod_{\langle i,j\rangle \in spans(S)} \mathrm{P}(\alpha_{ij}, x_{ij}|L_{ij}) \\ &= \prod_{\langle i,j\rangle} \mathrm{P}(\alpha_{ij}|L_{ij})\mathrm{P}(x_{ij}|L_{ij})\end{aligned}$$

The set of labels for constituent spans and distituent spans are forced to be disjoint, so $\mathrm{P}(L_{ij}|B_{ij})$ is given by

$$\mathrm{P}(L_{ij}|B_{ij}) = \begin{cases} 1 & \text{if } B_{ij} = \textit{false} \wedge L_{ij} = d \\ 0 & \text{if } B_{ij} = \textit{false} \wedge L_{ij} \neq d \\ 0 & \text{if } B_{ij} = \textit{true} \wedge L_{ij} = d \\ 1/|C-1| & \text{if } B_{ij} = \textit{true} \wedge L_{ij} \neq d \end{cases}$$

where d is a distinguished distituent-only label, and the other labels are sampled uniformly at each constituent span.

Intuitively, it seems that more classes should help, by allowing the system to distinguish different types of constituents and constituent contexts. However, it seemed to slightly hurt parsing accuracy overall. Figure 5.8 compares the performance for 2 versus 12 classes; in both cases, only one of the classes was allocated for distituents. Overall F_1 dropped very slightly with 12 classes, but the category recall numbers indicate that the errors shifted around substantially. PP accuracy is lower, which is not surprising considering that PPs

Classes	Tags	Precision	Recall	F_1	NP Recall	PP Recall	VP Recall	S Recall
2	Treebank	**63.8**	**80.2**	**71.1**	**83.4**	**78.5**	78.6	40.7
12	Treebank	63.6	80.0	70.9	82.2	59.1	82.8	57.0
2	Induced	56.8	71.1	63.2	52.8	56.2	**90.0**	**60.5**

Figure 5.8: Scores for the 2- and 12-class model with Treebank tags, and the 2-class model with induced tags.

Class 0		Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
NNP NNP	NN VBD	DT NN	NNP NNP	CD CD	VBN IN	MD VB	JJ NN
NN IN	NN NN	JJ NNS	NNP NNP NNP	CD NN	JJ IN	MD RB VB	JJ NNS
IN DT	NNS VBP	DT NNS	CC NNP	IN CD CD	DT NN	VBN IN	JJ JJ NN
DT JJ	NNS VBD	DT JJ NN	POS NN	CD NNS	JJ CC	WDT VBZ	CD NNS
NN VBZ	TO VB	NN NNS	NNP NNP NNP NNP	CD CD IN CD CD	DT JJ NN	JJ IN	NNP NN

Figure 5.9: Most frequent members of several classes found.

tend to appear rather optionally and in contexts in which other, easier categories also frequently appear. On the other hand, embedded sentence recall is substantially higher, possibly because of more effective use of the top-level sentences which occur in the context $\diamond$—$\diamond$.

The classes found, as might be expected, range from clearly identifiable to nonsense. Note that simply directly clustering all sequence types into 12 categories based on their local linear distributional signatures produced almost entirely the latter, with clusters representing various distituent types. Figure 5.9 shows several of the 12 classes. Class 0 is the model's distituent class. Its most frequent members are a mix of obvious distituents (IN DT, DT JJ, IN DT, NN VBZ) and seemingly good sequences like NNP NNP. However, there are many sequences of 3 or more NNP tags in a row, and not all adjacent pairs can possibly be constituents at the same time. Class 1 is mainly common NP sequences, class 2 is proper NPs, class 3 is NPs which involve numbers, and class 6 is $\overline{\text{N}}$ sequences, which tend to be linguistically right but unmarked in the treebank. Class 4 is a mix of seemingly good NPs, often from positions like VBZ–NN where they were *not* constituents, and other sequences that share such contexts with otherwise good NP sequences. This is a danger of not jointly modeling yield and context, and of not modeling any kind of recursive structure: our model cannot learn that a sequence is a constituent only in certain contexts (the best we can hope for is that such contexts will be learned as strong distituent contexts). Class 5 is mainly composed of verb phrases and verb groups. No class corresponded neatly to PPs: perhaps because they have no signature contexts. The 2-class model is effective at identifying them

only because they share contexts with a range of other constituent types (such as NPs and VPs).

5.3.3 Induced Parts-of-Speech

A reasonable criticism of the experiments presented so far, and some other earlier work, is that we assume treebank part-of-speech tags as input. This criticism could be two-fold. First, state-of-the-art supervised PCFGs do not perform nearly so well with their input delexicalized. We may be reducing data sparsity and making it easier to see a broad picture of the grammar, but we are also limiting how well we can possibly do. It is certainly worth exploring methods which supplement or replace tagged input with lexical input. However, we address here the more serious criticism: that our results stem from clues latent in the treebank tagging information which are conceptually posterior to knowledge of structure. For instance, some treebank tag distinctions, such as particle (RP) vs. preposition (IN) or predeterminer (PDT) vs. determiner (DT) or adjective (JJ), could be said to import into the tag set distinctions that can only be made syntactically.

To show results from a complete grammar induction system, we also did experiments starting with an automatic clustering of the words in the treebank (details in section 2.1.4. We do not believe that the quality of our tags matches that of the better methods of Schütze (1995), much less the recent results of Clark (2000). Nevertheless, using these tags as input still gave induced structure substantially above right-branching. Figure 5.8 shows the performance with induced tags compared to correct tags. Overall F_1 has dropped, but, interestingly, VP and S recall are higher. This seems to be due to a marked difference between the induced tags and the treebank tags: nouns are scattered among a disproportionately large number of induced tags, increasing the number of common NP sequences, but decreasing the frequency of each.

5.3.4 Convergence and Stability

A common issue with many previous systems is their sensitivity to initial choices. While the model presented here is clearly sensitive to the quality of the input tagging, as well as the qualitative properties of the initial completions, it does not suffer from the need to

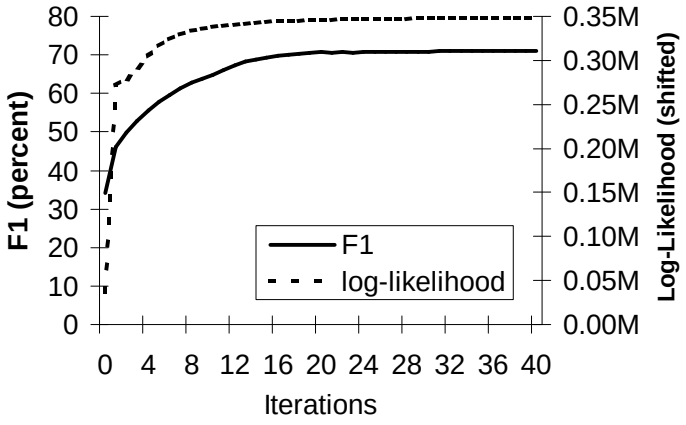


Figure 5.10: F_1 is non-decreasing until convergence.

inject noise to avoid an initial saddle point. Training on random subsets of the training data brought lower performance, but constantly lower over equal-size splits.

Figure 5.10 shows the overall F_1 score and the data likelihood according to our model during convergence.[6] Surprisingly, both are non-decreasing as the system iterates, indicating that data likelihood in this model corresponds well with parse accuracy.[7] Figure 5.12 shows recall for various categories by iteration. NP recall exhibits the more typical pattern of a sharp rise followed by a slow fall, but the other categories, after some initial drops, all increase until convergence.[8] These graphs stop at 40 iterations. The time to convergence varied according to smoothing amount, number of classes, and tags used, but the system almost always converged within 80 iterations, usually within 40.

5.3.5 Partial Supervision

For many practical applications, supplying a few gold parses may not be much more expensive than deploying a fully unsupervised system. To test the effect of partial supervision, we trained the CCM model on 90% of the WSJ10 corpus, and tested it on the remaining

[6]The data likelihood is not shown exactly, but rather we show the linear transformation of it calculated by the system (internal numbers were scaled to avoid underflow).

[7]Pereira and Schabes (1992) find otherwise for PCFGs.

[8]Models in the next chapter also show good correlation between likelihood and evaluation metrics, but generally not monotonic as in the present case.

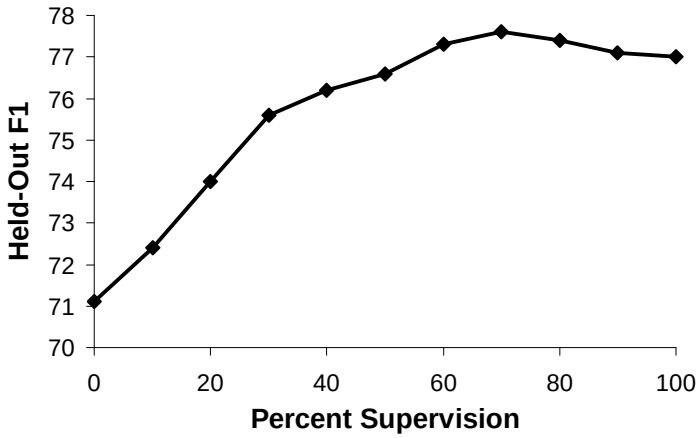


Figure 5.11: Partial supervision

10%. Various fractions of that 90% were labeled with their gold treebank parses; during the learning phase, analyses which crossed the brackets of the labeled parses were given zero weight (but the CCM still filled in binary analyses inside flat gold trees). Figure 5.11 shows F_1 on the held-out 10% as supervision percent increased. Accuracy goes up initially, though it drops slightly at very high supervision levels. The most interesting conclusion from this graph is that small amounts of supervision do not actually seem to help the CCM very much, at least when used in this naive fashion.

5.3.6 Details

There are several details necessary to get good performance out of this model.

Initialization

The completions in this model, just as in the inside-outside algorithm for PCFGs, are distributions over trees. For natural language trees, these distributions are very non-uniform. Figure 5.13 shows empirical bracketing distributions for three languages. These distributions show, over treebank parses of 10-word sentences, the fraction of trees with a constituent over each start and end point. On the other hand, figure 5.14 (b) shows the bracket fractions in a distribution which puts equal weight on each (unlabeled) binary tree. The

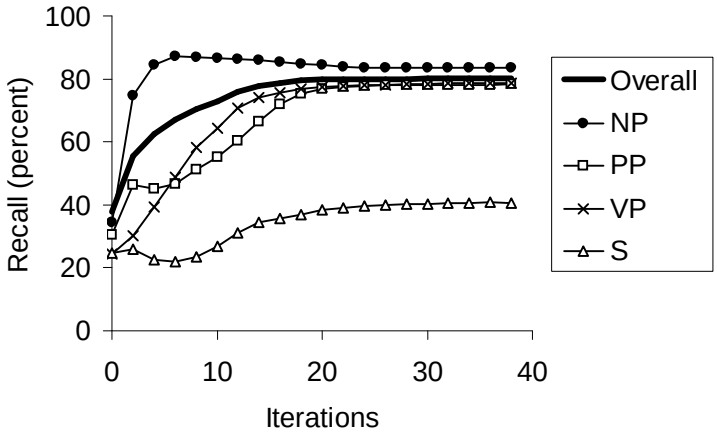
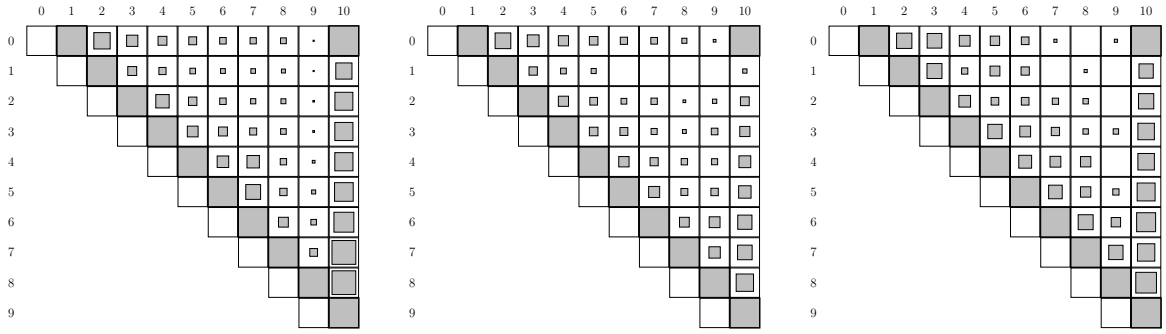


Figure 5.12: Recall by category during convergence.

most important difference between the actual and tree-uniform bracketing distributions is that uniform trees are dramatically more likely to have central constituents, while in natural language constituents tend to either start at the beginning of a sentence or end at the end of the sentence.

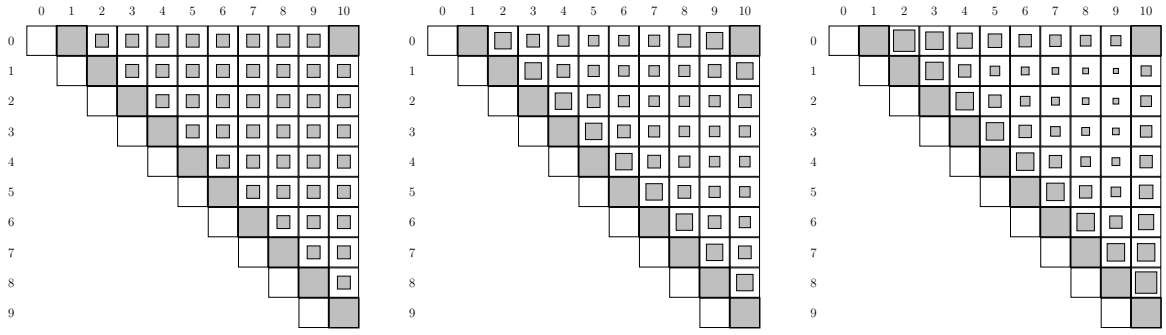
What this means for an induction algorithm is important. Most "uniform" grammars, such as a PCFG in which all rewrites have equal weight, or our current proposal with the constituent and context multinomials being uniform, will have the property that all trees will receive equal scores (or roughly so, modulo any initial perturbation). Therefore, if we begin with an E-step using such a grammar, most first M-steps will be presented with a posterior that looks like figure 5.14(b). If we have a better idea about what the posteriors should look like, we can begin with an E-step instead, such as the one where all non-trivial brackets are equally likely, shown in figure 5.14(a) (this bracket distribution does not correspond to any distribution over binary trees).

Now, we don't necessarily know what the posterior should look like, and we don't want to bias it too much towards any particular language. However, we found that another relatively neutral distribution over trees made a good initializer. In particular, consider the following uniform-splitting process of generating binary trees over k terminals: choose a split point at random, then recursively build trees by this process on each side of the



(a) English (b) German (c) Chinese

Figure 5.13: Empirical bracketing distributions for 10-word sentences in three languages (see chapter 2 for corpus descriptions).



(a) Uniform over Brackets (b) Uniform over Trees (c) Uniform over Splits

Figure 5.14: Bracketing distributions for several notions of "uniform": all brackets having equal likelihood, all trees having equal likelihood, and all recursive splits having equal likelihood.

Initialization	Precision	Recall	F_1	CB
Tree Uniform	55.5	70.5	62.1	1.58
Bracket Uniform	55.6	70.6	62.2	1.57
Split Uniform	64.7	82.2	72.4	0.99
Empirical	65.5	83.2	73.3	1.00

Figure 5.15: CCM performance on WSJ10 as the initializer is varied. Unlike other numbers in this chapter, these values are micro-averaged at the bracket level, as is typical for supervised evaluation, and give credit for the whole-sentence bracket).

split. This process gives a distribution $\mathrm{P}_{\mathrm{split}}$ which puts relatively more weight on unbalanced trees, but only in a very general, non language-specific way. The posterior of the split-uniform distribution is shown in figure 5.14 (c). Another useful property of the split distribution is that it can be calculated in closed form (details in appendix B.2).

In figure 5.13, aside from the well-known right-branching tendency of English (and Chinese), a salient characteristic of all three languages is that central brackets are relatively rare. The split-uniform distribution also shows this property, while the bracket-uniform distribution and the "natural" tree-uniform distribution do not. Unsurprisingly, results when initializing with the bracket-uniform and tree-uniform distributions were substantially worse than using the split-uniform one. Using the actual posterior was, interestingly, only slightly better (see figure 5.15).

While the split distribution was used as an initial completion, it was not used in the model itself. It seemed to bias too strongly against balanced structures, and led to entirely linear-branching structures.

Smoothing

The smoothing used was straightforward, but very important. For each yield α or context x, we added 10 counts of that item: 2 as a constituent and 8 as a distituent. This reflected the relative skew of random spans being more likely to be distituents.

Sentence Length

A weakness of the current model is that it performs much better on short sentences than longer ones: F_1 drops all the way to 53.4% on sentences of length up to 15 (see figure 6.9 in section 6.3). One likely cause is that as spans get longer, span type counts get smaller, and so the parsing is driven by the less-informative context multinomials. Indeed, the primary strength of this system is that it chunks simple NP and PP groups well; longer sentences are less well-modeled by linear spans and have more complex constructions: relative clauses, coordination structures, and so on. The improved models in chapter 6 degrade substantially less with increased sentence length (section 6.3).

5.4 Conclusions

We have presented a simple generative model for the unsupervised distributional induction of hierarchical linguistic structure. The system achieves the above-baseline unsupervised parsing scores on the WSJ10 and ATIS data sets. The induction algorithm combines the benefits of EM-based parameter search and distributional clustering methods. We have shown that this method acquires a substantial amount of correct structure, to the point that the most frequent discrepancies between the induced trees and the treebank gold standard are systematic alternate analyses, many of which are linguistically plausible. We have shown that the system is not overly reliant on supervised POS tag input, and demonstrated increased accuracy, speed, simplicity, and stability compared to previous systems.

Chapter 6

Dependency Models

6.1 Unsupervised Dependency Parsing

Most recent work (and progress) in unsupervised parsing has come from tree or phrase-structure based models, but there are compelling reasons to reconsider unsupervised *dependency* parsing as well. First, most state-of-the-art *supervised* parsers make use of specific lexical information in addition to word-class level information – perhaps lexical information could be a useful source of information for unsupervised methods. Second, a central motivation for using tree structures in computational linguistics is to enable the extraction of dependencies – function-argument and modification structures – and it might be more advantageous to induce such structures directly. Third, as we show below, for languages such as Chinese, which have few function words, and for which the definition of lexical categories is much less clear, dependency structures may be easier to detect.

6.1.1 Representation and Evaluation

An example dependency representation of a short sentence is shown in figure 6.1(a), where, following the traditional dependency grammar notation, the regent or head of a dependency is marked with the tail of the dependency arrow, and the dependent is marked with the arrowhead (Mel′čuk 1988). It will be important in what follows to see that such a representation is isomorphic (in terms of strong generative capacity) to a restricted form of phrase

structure grammar, where the set of terminals and nonterminals is identical, and every rule is of the form X → X Y or X → Y X (Miller 1999), giving the isomorphic representation of figure 6.1(a) shown in figure 6.1(b).[1] Depending on the model, part-of-speech categories may be included in the dependency representation, as suggested here, or dependencies may be directly between words (bilexical dependencies). Below, we will assume an additional reserved nonterminal ROOT, whose sole dependent is the head of the sentence. This simplifies the notation, math, and the evaluation metric.

A dependency analysis will always consist of exactly as many dependencies as there are words in the sentence. For example, in the dependency structure of figure 6.1(b), the dependencies are {(ROOT, *fell*), (*fell, payrolls*), (*fell, in*), (*in, September*), (*payrolls, Factory*)}. The quality of a hypothesized dependency structure can hence be evaluated by accuracy as compared to a gold-standard dependency structure, by reporting the percentage of dependencies shared between the two analyses.

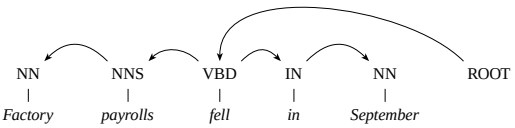
It is important to note that the Penn treebanks do *not* include dependency annotations; however, the automatic dependency rules from (Collins 1999) are sufficiently accurate to be a good benchmark for unsupervised systems for the time being (though see below for specific issues). Similar head-finding rules were used for Chinese experiments. The NEGRA corpus, however, does supply hand-annotated dependency structures.

Where possible, we report an accuracy figure for both directed and undirected dependencies. Reporting undirected numbers has two advantages: first, it facilitates comparison with earlier work, and, more importantly, it allows one to partially obscure the effects of alternate analyses, such as the systematic choice between a modal and a main verb for the head of a sentence (in either case, the two verbs would be linked, but the direction would vary).

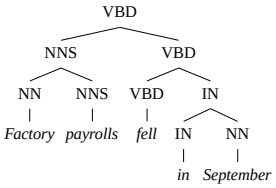
6.1.2 Dependency Models

The dependency induction task has received relatively little attention; the best known work is Carroll and Charniak (1992), Yuret (1998), and Paskin (2002). All systems that we are

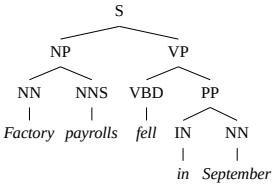
[1] Strictly, such phrase structure trees are isomorphic not to flat dependency structures, but to specific derivations of those structures which specify orders of attachment among multiple dependents which share a common head.



(a) Classical Dependency Structure



(b) Dependency Structure as CF Tree



(c) CFG Structure

Figure 6.1: Three kinds of parse structures.

Figure 6.2: Dependency graph with skeleton chosen, but words not populated.

aware of operate under the assumption that the probability of a dependency structure is the product of the scores of the dependencies (attachments) in that structure. Dependencies are seen as ordered (head, dependent) pairs of words, but the score of a dependency can optionally condition on other characteristics of the structure, most often the direction of the dependency (whether the arrow points left or right).

Some notation before we present specific models: a dependency d is a pair $\langle h, a \rangle$ of a head and an argument, which are words in a sentence s, in a corpus S. For uniformity of notation with chapter 5, words in s are specified as size-one spans of s: for example the first word would be ${}_0s_1$. A dependency structure D over a sentence is a set of dependencies (arcs) which form a planar, acyclic graph rooted at the special symbol ROOT, and in which each word in s appears as an argument exactly once. For a dependency structure D, there is an associated graph G which represents the number of words and arrows between them, without specifying the words themselves (see figure 6.2). A graph G and sentence s together thus determine a dependency structure. The dependency structure is the object generated by all of the models that follow; the steps in the derivations vary from model to

model.

Existing generative dependency models intended for unsupervised learning have chosen to first generate a word-free graph G, then populate the sentence s conditioned on G. For instance, the model of Paskin (2002), which is broadly similar to the semi-probabilistic model in Yuret (1998), first chooses a graph G uniformly at random (such as figure 6.2), then fills in the words, starting with a fixed root symbol (assumed to be at the rightmost end), and working down G until an entire dependency structure D is filled in (figure 6.1a). The corresponding probabilistic model is

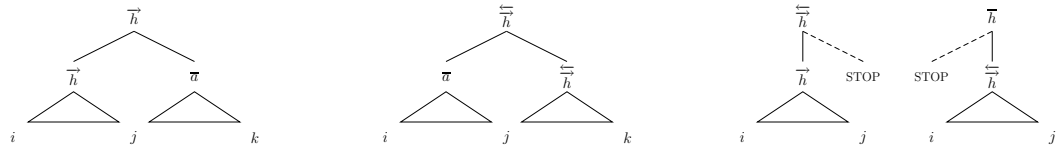
$$\begin{aligned} \mathrm{P}(D) &= \mathrm{P}(s, G) \\ &= \mathrm{P}(G)\mathrm{P}(s|G) \\ &= \mathrm{P}(G) \prod_{(i,j,dir)\in G} \mathrm{P}({}_{i-1}s_i|_{j-1}s_j, dir)\,. \end{aligned}$$

In Paskin (2002), the distribution $\mathrm{P}(G)$ is fixed to be uniform, so the only model parameters are the conditional multinomial distributions $\mathrm{P}(a|h, dir)$ that encode which head words take which other words as arguments. The parameters for left and right arguments of a single head are completely independent, while the parameters for first and subsequent arguments in the same direction are identified.

In those experiments, the model above was trained on over 30M words of raw newswire, using EM in an entirely unsupervised fashion, and at great computational cost. However, as shown in figure 6.3, the resulting parser predicted dependencies at below chance level (measured by choosing a random dependency structure). This below-random performance seems to be because the model links word pairs which have high mutual information (such as occurrences of *congress* and *bill*) regardless of whether they are plausibly syntactically related. In practice, high mutual information between words is often stronger between two topically similar nouns than between, say, a preposition and its object (worse, it's also usually stronger between a verb and a selected preposition than that preposition and its object).

Model	Dir.	Undir.
English (WSJ)		
Paskin 01		39.7
RANDOM		41.7
Charniak and Carroll 92-inspired		44.7
ADJACENT		53.2
DMV		54.4
English (WSJ10)		
RANDOM	30.1	45.6
ADJACENT	33.6	56.7
DMV	43.2	63.7
German (NEGRA10)		
RANDOM	21.8	41.5
ADJACENT	32.6	51.2
DMV	36.3	55.8
Chinese (CTB10)		
RANDOM	35.9	47.3
ADJACENT	30.2	47.3
DMV	42.5	54.2

Figure 6.3: Parsing performance (directed and undirected dependency accuracy) of various dependency models on various treebanks, along with baselines.



(a) (b) (c) (d)

Figure 6.4: Dependency configurations in a lexicalized tree: (a) right attachment, (b) left attachment, (c) right stop, (d) left stop. h and a are head and argument words, respectively, while i, j, and k are positions between words. Not show is the step (if modeled) where the head chooses to generate right arguments before left ones, or the configurations if left arguments are to be generated first.

The specific connection which argues why this model roughly learns to maximize mutual information is that in maximizing

$$\mathrm{P}(D) = \mathrm{P}(G) \prod_{(i,j,dir) \in G} \mathrm{P}(_{i-1}s_i|_{j-1}s_j, dir)$$

it is also maximizing

$$\frac{\mathrm{P}(D)}{\mathrm{P}'(D)} = \frac{\mathrm{P}(G) \prod_{(i,j,dir) \in G} \mathrm{P}(_{i-1}s_i|_{j-1}s_j, dir)}{\mathrm{P}(G) \prod_i \mathrm{P}(_{i-1}s_i)}$$

which, dropping the dependence on directionality, gives

$$\begin{aligned} \frac{\mathrm{P}(D)}{\mathrm{P}'(D)} &= \frac{\mathrm{P}(G) \prod_{(i,j) \in G} \mathrm{P}(_{i-1}s_i|_{j-1}s_j)}{\mathrm{P}(G) \prod_i \mathrm{P}(_{i-1}s_i)} \\ &= \prod_{(i,j) \in G} \frac{\mathrm{P}(_{i-1}s_{i,j-1}\, s_j)}{\mathrm{P}(_{i-1}s_i)\mathrm{P}(_{j-1}s_j)} \end{aligned}$$

which is a product of (pointwise) mutual information terms.

One might hope that the problem with this model is that the actual lexical items are too semantically charged to represent workable units of syntactic structure. If one were to apply the Paskin (2002) model to dependency structures parameterized simply on the word-classes, the result would be isomorphic to the "dependency PCFG" models described in Carroll and Charniak (1992) (see section 5.1). In these models, Carroll and Charniak considered PCFGs with precisely the productions (discussed above) that make them isomorphic to dependency grammars, with the terminal alphabet being simply parts-of-speech. Here, the rule probabilities are equivalent to $\mathrm{P}(\mathrm{Y}|\mathrm{X}, \textit{right})$ and $\mathrm{P}(\mathrm{Y}|\mathrm{X}, \textit{left})$ respectively.[2] The actual experiments in Carroll and Charniak (1992) do not report accuracies that we can compare to, but they suggest that the learned grammars were of extremely poor quality. As discussed earlier, a main issue in their experiments was that they randomly initialized the production (attachment) probabilities. As a result, their learned grammars were of very

[2]There is another, more subtle distinction: in the Paskin work, a canonical ordering of multiple attachments was fixed, while in the Carroll and Charniak work all attachment orders are considered to be different (equal scoring) structures when listing analyses, giving a relative bias in the Carroll and Charniak work towards structures where heads take more than one argument.

poor quality and had high variance. However, one nice property of their structural constraint, which all dependency models share, is that the symbols in the grammar are not symmetric. Even with a grammar in which the productions are initially uniform, a symbol X can only possibly have non-zero posterior likelihood over spans which contain a matching terminal X. Therefore, one can start with uniform rewrites and let the interaction between the data and the model structure break the initial symmetry. If one recasts their experiments in this way, they achieve an accuracy of 44.7% on the Penn treebank, which is higher than choosing a random dependency structure, but lower than simply linking all adjacent words into a left-headed (and right-branching) structure (53.2%). That this should outperform the bilexical model is in retrospect unsurprising: a major source of non-syntactic information has been hidden from the model, and accordingly there is one fewer unwanted trend that might be detected in the process of maximizing data likelihood.

A huge limitation of both of the above models, however, is that they are incapable of encoding even first-order valence facts, valence here referring in a broad way to the regularities in number and type of arguments a word or word class takes (i.e., including but not limited to subcategorization effects). For example, the former model will attach all occurrences of "new" to "york," even if they are not adjacent, and the latter model learns that nouns to the left of the verb (usually subjects) attach to the verb. But then, given a NOUN NOUN VERB sequence, both nouns will attach to the verb – there is no way that the model can learn that verbs have exactly one subject. We now turn to an improved dependency model that addresses this problem.

6.2 An Improved Dependency Model

The dependency models discussed above are distinct from dependency models used inside high-performance supervised probabilistic parsers in several ways. First, in supervised models, a head outward process is modeled (Eisner 1996, Collins 1999). In such processes, heads generate a sequence of arguments outward to the left or right, conditioning on not only the identity of the head and direction of the attachment, but also on some notion of distance or valence. Moreover, in a head-outward model, it is natural to model stop steps, where the final argument on each side of a head is always the special symbol STOP. Models

like Paskin (2002) avoid modeling STOP by generating the graph skeleton G first, uniformly at random, then populating the words of s conditioned on G. Previous work (Collins 1999) has stressed the importance of including termination probabilities, which allows the graph structure to be generated jointly with the terminal words, precisely because it does allow the modeling of required dependents.

We propose a simple head-outward dependency model over word classes which includes a model of valence, which we call *DMV* (for *dependency model with valence*). We begin at the ROOT. In the standard way (see below), each head generates a series of non-STOP arguments to one side, then a STOP argument to that side, then non-STOP arguments to the other side, then a second STOP.

For example, in the dependency structure in figure 6.1, we first generate a single child of ROOT, here *fell*. Then we recurse to the subtree under *fell*. This subtree begins with generating the right argument *in*. We then recurse to the subtree under *in* (generating *September* to the right, a right STOP, and a left STOP). Since there are no more right arguments after *in*, its right STOP is generated, and the process moves on to the left arguments of *fell*.

In this process, there are two kinds of derivation events, whose local probability factors constitute the model's parameters. First, there is the decision at any point whether to terminate (generate STOP) or not: $\mathrm{P}_{\text{STOP}}(\text{STOP}|h, dir, adj)$. This is a binary decision conditioned on three things: the head h, the direction (generating to the left or right of the head), and the adjacency (whether or not an argument has been generated yet in the current direction, a binary variable). The stopping decision is estimated directly, with no smoothing. If a stop is generated, no more arguments are generated for the current head to the current side. If the current head's argument generation does not stop, another argument is chosen using: $\mathrm{P}_{\text{CHOOSE}}(a|h, dir)$. Here, the argument is picked conditionally on the identity of the head (which, recall, is a word class) and the direction. This term, also, is not smoothed in any way. Adjacency has no effect on the identity of the argument, only on the likelihood of termination. After an argument is generated, its subtree in the dependency structure is recursively generated.

This process should be compared to what is generally done in supervised parsers (Collins 1999, Charniak 2000, Klein and Manning 2003). The largest difference is that supervised parsers condition actions on the identity of the head word itself. The lexical identity is a

good feature to have around in a supervised system, where syntactic lexical facts can be learned effectively. In our unsupervised experiments, having lexical items in the model led to distant topical associations being preferentially modeled over class-level syntactic patterns, though it would clearly be advantageous to discover a mechanism for acquiring the richer kinds of models used in the supervised case. Supervised parsers' decisions to stop or continue generating arguments are also typically conditioned on finer notions of distance than adjacent/non-adjacent (buckets or punctuation-defined distance). Moreover, decisions about argument identity are conditioned on the identity of previous arguments, not just a binary indicator of whether there were any previous ones. This richer history allows for the explicit modeling of inter-argument correlations, such as subcategorization/selection preferences and argument ordering trends. Again, for the unsupervised case, this much freedom can be dangerous. We did not use such richer histories, their success in supervised systems suggests that they could be exploited here, perhaps in a system which originally ignored richer context, then gradually began to model it.

Formally, for a dependency structure D, let each word h have left dependents $deps_D(h, l)$ and right dependents $deps_D(h, r)$. The following recursion defines the probability of the fragment $D(h)$ of the dependency tree rooted at h:

$$\begin{aligned}\mathrm{P}(D(h)) = \quad & \prod_{dir \in \{l,r\}} \prod_{a \in deps_D(h,dir)} \mathrm{P}_{\text{STOP}}(\neg\text{STOP}|h, dir, adj) \\ & \mathrm{P}_{\text{CHOOSE}}(a|h, dir)\mathrm{P}(D(a)) \\ & \mathrm{P}_{\text{STOP}}(\text{STOP}|h, dir, adj)\end{aligned}$$

One can view a structure generated by this derivational process as a "lexicalized" tree composed of the local binary and unary context-free configurations shown in figure 6.4.[3] Each configuration equivalently represents either a head-outward derivation step or a context-free rewrite rule. There are four such configurations. Figure 6.4(a) shows a head h taking a right argument a. The tree headed by h contains h itself, possibly some right

[3]It is lexicalized in the sense that the labels in the tree are derived from terminal symbols, but in our experiments the terminals were word classes, not individual lexical items.

arguments of h, but no left arguments of h (they attach after all the right arguments). The tree headed by a contains a itself, along with all of its left and right children. Figure 6.4(b) shows a head h taking a left argument a – the tree headed by h must have already generated its right stop to do so. Figure 6.4(c) and figure 6.4(d) show the *sealing* operations, where STOP derivation steps are generated. The left and right marks on node labels represent left and right STOPs that have been generated.[4]

The basic inside-outside algorithm (Baker 1979) can be used for re-estimation. For each sentence $s \in S$, it gives us $c_s(x : i, j)$, the expected fraction of parses of s with a node labeled x extending from position i to position j. The model can be re-estimated from these counts. For example, to re-estimate an entry of $\mathrm{P}_{\text{STOP}}(\text{STOP}|h, \textit{left}, \textit{non-adj})$ according to a current model Θ, we calculate two quantities.[5] The first is the (expected) number of trees headed by $\overleftrightarrow{h}$ whose start position i is strictly left of h. The second is the number of trees headed by $\overline{h}$ with start position i strictly left of h. The ratio is the MLE of that local probability factor:

$$\mathrm{P}_{\text{STOP}}(\text{STOP}|h, \textit{left}, \textit{non-adj}) = \frac{\sum_{s \in S} \sum_{i < loc(h)} \sum_k c(\overline{h} : i, k)}{\sum_{s \in S} \sum_{i < loc(h)} \sum_k c(\overleftrightarrow{h} : i, k)}$$

This can be intuitively thought of as the relative number of times a tree headed by h had already taken at least one argument to the left, had an opportunity to take another, but didn't.[6] Section A.2 has a more detailed exposition of the details of calculating the necessary expectations.

Initialization is important to the success of any local search procedure. As in chapter 5, we chose to initialize EM not with an initial model, but with an initial guess at posterior distributions over dependency structures (completions). For the first-round, we constructed

[4]Note that the asymmetry of the attachment rules enforces the right-before-left attachment convention. This is harmless and arbitrary as far as dependency evaluations go, but imposes an X-bar-like structure on the constituency assertions made by this model. This bias/constraint is dealt with in section 6.3.

[5]To simplify notation, we assume each word h occurs at most one time in a given sentence, between indexes $loc(h)$ and $loc(h) + 1$.

[6]As a final note, in addition to enforcing the right-argument-first convention, we constrained ROOT to have at most a single dependent, by a similar device.

a somewhat ad-hoc "harmonic" completion where all non-ROOT words took the same number of arguments, and each took other words as arguments in inverse proportion to (a constant plus) the distance between them. The ROOT always had a single argument and took each word with equal probability. This structure had two advantages: first, when testing multiple models, it is easier to start them all off in a common way by beginning with an M-step, and, second, it allowed us to point the model in the vague general direction of what linguistic dependency structures should look like. It should be emphasized that this initialization was important in getting reasonable patterns out of this model.

On the WSJ10 corpus, the DMV model recovers a substantial fraction of the broad dependency trends: 43.2% of guessed directed dependencies were correct (63.7% ignoring direction). To our knowledge, this is the first published result to break the adjacent-word heuristic (at 33.6% for this corpus). Verbs are the sentence heads, prepositions take following noun phrases as arguments, adverbs attach to verbs, and so on. Figure 6.5 shows the most frequent discrepancies between the test dependencies and the model's guesses. Most of the top mismatches stem from the model systematically choosing determiners to be the heads of noun phrases, where the test trees have the rightmost noun as the head. The model's choice is supported by a good deal of linguistic research (Abney 1987), and is sufficiently systematic that we also report the scores where the NP headship rule is changed to percolate determiners when present. On this adjusted metric, the score jumps hugely to 55.7% directed (and 67.9% undirected). There are other discrepancy types, such as modals dominating main verbs, choice of the wrong noun as the head of a noun cluster, and having some sentences headed by conjunctions.

This model also works on German and Chinese at above-baseline levels (55.8% and 54.2% undirected, respectively), with no modifications whatsoever. In German, the largest source of errors is also the systematic postulation of determiner-headed noun-phrases. The second largest source is that adjectives are (incorrectly) considered to be the head in adjective-noun units. The third source is the incorrect attachment of adjectives into determiners inside definite NPs. Mistakes involving verbs and other classes are less common, but include choosing past participles rather than auxiliaries as the head of periphrastic verb constructions. In Chinese, there is even more severe confusion inside nominal sequences, possibly because the lack of functional syntax makes the boundaries between adjacent NPs

English using DMV			
Overproposals		Underproposals	
DT ← NN	3083	DT → NN	3079
NNP ← NNP	2108	NNP → NNP	1899
CC → ROOT	1003	IN ← NN	779
IN ← DT	858	DT → NNS	703
DT ← NNS	707	NN → VBZ	688
MD → VB	654	NN ← IN	669
DT → IN	567	MD ← VB	657
DT → VBD	553	NN → VBD	582
TO → VB	537	VBD ← NN	550
DT → VBZ	497	VBZ ← NN	543

English using CCM+DMV			
Overproposals		Underproposals	
DT ← NN	3474	DT → NN	3079
NNP ← NNP	2096	NNP → NNP	1898
CD → CD	760	IN ← NN	838
IN ← DT	753	NN → VBZ	714
DT ← NNS	696	DT → NNS	672
DT → IN	627	NN ← IN	669
DT → VBD	470	CD ← CD	638
DT → VBZ	420	NN → VBD	600
NNP → ROOT	362	VBZ ← NN	553
NNS → IN	347	VBD ← NN	528

Figure 6.5: Dependency types most frequently overproposed and underproposed for English, with the DMV alone and with the combination model.

unclear. For example, temporal nouns often take adjacent proper nouns as arguments – all other classes of errors are much less salient.

This dependency induction model is reasonably successful. However, the model can be substantially improved by paying more attention to syntactic constituency, at the same time as modeling dependency structure. To this end, we next present a combined model that exploits kinds of structure. As we will see, the combined model finds correct dependencies more successfully than the model above, and finds constituents more successfully than the model of chapter 5.

6.3 A Combined Model

The CCM and the DMV models have a common ground. Both can be seen as models over lexicalized trees composed of the configurations in figure 6.4. For the DMV, it is already a model over these structures. At the "attachment" rewrite for the CCM in (a/b), we assign the quantity:

$$\frac{\mathrm{P}({}_i s_k|\textit{true})\mathrm{P}({}_{i-1}s_i \sim {}_k s_{k+1}|\textit{true})}{\mathrm{P}({}_i s_k|\textit{false})\mathrm{P}({}_{i-1}s_i \sim {}_k s_{k+1}|\textit{false})}$$

which is the odds ratio of generating the subsequence and context for span $\langle i, k\rangle$ as a constituent as opposed to as a non-constituent. If we multiply all trees' attachment scores by

$$\prod_{\langle i,j\rangle} \mathrm{P}({}_i s_j|\textit{false})\mathrm{P}({}_{i-1}s_i \sim {}_j s_{j+1}|\textit{false})$$

the denominators of the odds ratios cancel, and we are left with each tree being assigned the probability it would have received under the CCM.[7]

In this way, both models can be seen as generating either constituency or dependency structures. Of course, the CCM will generate fairly random dependency structures (constrained only by bracketings). Getting constituency structures from the DMV is also problematic, because the choice of which side to first attach arguments on has ramifications on constituency – it forces x-bar-like structures – even though it is an arbitrary convention as

[7]This scoring function as described is not a generative model over lexicalized trees, because it has no generation step at which nodes' lexical heads are chosen. This can be corrected by multiplying in a "head choice" factor of $1/(k-j)$ at each final "sealing" configuration (d). In practice, this correction factor was harmful for the model combination, since it duplicated a strength of the dependency model, badly.

far as dependency evaluations are concerned. For example, if we attach right arguments first, then a verb with a left subject and a right object will attach the object first, giving traditional VPs, while the other attachment order gives subject-verb groups. To avoid this bias, we alter the DMV in the following ways. When using the dependency model alone, we allow each word to have even probability for either generation order, this order being chosen as the first step in a head's outward dependency generation process (in each actual head derivation, only one order occurs). When using the models together, better performance was obtained by releasing the one-side-attaching-first requirement entirely.[8]

In figure 6.6, we give the behavior of the CCM constituency model and the DMV dependency model on both constituency and dependency induction. Unsurprisingly, their strengths are complementary. The CCM is better at recovering constituency (except for Chinese where neither is working particularly well), and the dependency model is better at recovering dependency structures. It is reasonable to hope that a combination model might exhibit the best of both. In the supervised parsing domain, for example, scoring a lexicalized tree with the product of a simple lexical dependency model and a PCFG model can outperform each factor on its respective metric (Klein and Manning 2003).

In the combined model, we score each tree with the product of the probabilities from the individual models above. We use the inside-outside algorithm to sum over all lexicalized trees, similarly to the situation in section 6.2. The tree configurations are shown in figure 6.4. For each configuration, the relevant scores from each model are multiplied together. For example, consider figure 6.4(a). From the CCM we generate ${}_i s_k$ as a constituent and its corresponding context. From the dependency model, we pay the cost of h taking a as a right argument ($\mathrm{P}_{\text{CHOOSE}}$), as well as the cost of not stopping (P_{STOP}). The other configurations are similar. We then run the inside-outside algorithm over this product model. From the results, we can extract the statistics needed to re-estimate both individual models.[9]

The models in combination were initialized in the same way as when they were run individually. Sufficient statistics were separately taken off these individual completions. From then on, the resulting models were used together during re-estimation.

[8]With no one-side-first constraint, the proper derivation process chooses whether to stop entirely before each dependent, and if not choose a side to generate on, then generate an argument to that side.

[9]The product, like the CCM itself, is mass-deficient.

	Constituency			Dependency	
Model	UP	UR	UF_1	Dir	Undir
English (WSJ10 – 7422 Sentences)					
LBRANCH/RHEAD	25.6	32.6	28.7	33.6	56.7
RANDOM	31.0	39.4	34.7	30.1	45.6
RBRANCH/LHEAD	55.1	70.0	61.7	24.0	55.9
DMV	46.6	59.2	52.1	43.2	62.7
CCM	64.2	81.6	71.9	23.8	43.3
DMV+CCM (POS)	**69.3**	**88.0**	**77.6**	**47.5**	**64.5**
DMV+CCM (DISTR.)	65.2	82.8	72.9	42.3	60.4
UBOUND	78.8	100.0	88.1	100.0	100.0
German (NEGRA10 – 2175 Sentences)					
LBRANCH/RHEAD	27.4	48.8	35.1	32.6	51.2
RANDOM	27.9	49.6	35.7	21.8	41.5
RBRANCH/LHEAD	33.8	60.1	43.3	21.0	49.9
DMV	38.4	69.5	49.5	40.0	57.8
CCM	48.1	85.5	61.6	25.5	44.9
DMV+CCM	**49.6**	**89.7**	**63.9**	**50.6**	**64.7**
UBOUND	56.3	100.0	72.1	100.0	100.0
Chinese (CTB10 – 2437 Sentences)					
LBRANCH/RHEAD	26.3	48.8	34.2	30.2	43.9
RANDOM	27.3	50.7	35.5	35.9	47.3
RBRANCH/LHEAD	29.0	53.9	37.8	14.2	41.5
DMV	**35.9**	**66.7**	**46.7**	42.5	54.2
CCM	34.6	64.3	45.0	23.8	40.5
DMV+CCM	33.3	62.0	43.3	**55.2**	**60.3**
UBOUND	53.9	100.0	70.1	100.0	100.0

Figure 6.6: Parsing performance of the combined model on various treebanks, along with baselines.

Figure 6.6 summarizes the results. The combined model beats the CCM on English F_1: 77.6 vs. 71.9. To give a concrete indication of how the constituency analyses differ with and without the addition of the dependency model, figure 6.7 shows the sequences which were most frequently overproposed and underproposed as constituents, as well as the crossing proposals, which are the overproposals which actually cross a gold bracket. Note that in the combination model, verb groups disappear and adverbs are handled more correctly (this can be seen in both mistake summaries).

Figure 6.6 also shows the combination model's score when using word classes which were induced entirely automatically, using the same induced classes as in chapter 5. These classes show some degradation, e.g. 72.9 F_1, but it is worth noting that these totally unsupervised numbers are better than the performance of the CCM model running off of Penn treebank word classes. Again, if we modify the gold standard so as to make determiners the head of NPs, then this model with distributional tags scores even better with 50.6% on directed and 64.8% on undirected dependency accuracy.

On the German data, the combination again outperforms each factor alone, though while the combination was most helpful at boosting constituency quality for English, for German it provided a larger boost to the dependency structures. Figure 6.8 shows the common mistake sequences for German, with and without the DMV component. The most dramatic improvement is the more consistent use of verb-object VPs instead of subject-verb groups. Note that for the German data, the gold standard is extremely flat. This is why the precision is so low (49.6% in the combination model) despite the rather high recall (89.7%): in fact the crossing bracket rate is extremely low (0.39, cf. 0.68 for the English combination model).

Finally, on the Chinese data, the combination did substantially boost dependency accuracy over either single factor, but actually suffered a small drop in constituency.[10] Overall, the combination is able to combine the individual factors in an effective way.

To point out one final advantage of the combined model over the CCM (though not the DMV), consider figure 6.9, which shows how the accuracy of the combined model degrades with longer maximum sentence length (10 being WSJ10). On the constituency F_1 measure,

[10]This seems to be partially due to the large number of unanalyzed fragments in the Chinese gold standard, which leave a very large fraction of the posited bracketings completely unjudged.

English using CCM					
Overproposals		Underproposals		Crossing	
JJ NN	1022	NNP NNP	183	MD VB	418
NNP NNP	453	TO CD CD	159	RB VB	306
MD VB	418	NNP POS	159	IN NN	192
DT NN	398	NN NNS	140	POS NN	172
RB VB	349	NN NN	101	CD CD IN CD CD	154
JJ NNS	320	CD CD	72	MD RB VB	148
NNP NN	227	IN CD	69	RB VBN	103
RB VBN	198	TO VB	66	CD NNS	80
IN NN	196	RB JJ	63	VNB TO	72
POS NN	172	IN NNP	62	NNP RB	66

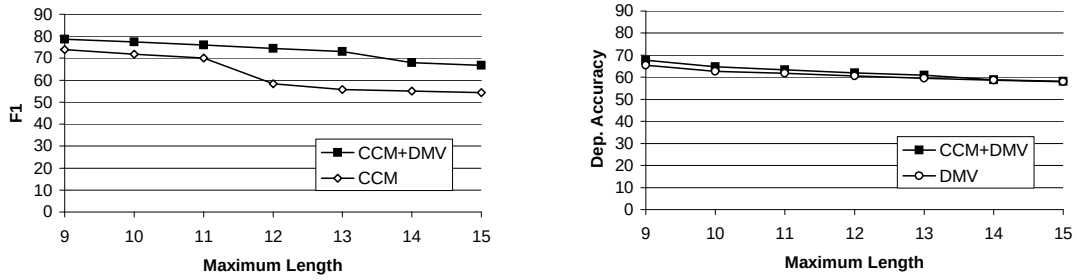
English using CCM+DMV					
Overproposals		Underproposals		Crossing	
JJ NN	1022	NNP NNP	167	CD CD IN CD CD	154
NNP NNP	447	TO CD CD	154	NNS RB	133
DT NN	398	IN NN	76	NNP NNP NNP	67
JJ NNS	294	IN DT NN	65	JJ NN	66
NNP NN	219	IN CD	60	NNP RB	59
NNS RB	164	CD NNS	56	NNP NNP NNP NNP	51
NNP NNP NNP	156	NNP NNP NNP	54	NNP NNP	50
CD CD IN CD CD	155	IN NNP	54	NNS DT NN	41
TO CD CD IN CD CD	154	NN NNS	49	IN PRP	41
CD NN TO CD CD IN CD CD	120	RB JJ	47	RB PRP	33

Figure 6.7: Sequences most frequently overproposed, underproposed, and proposed in locations crossing a gold bracket for English, for the CCM and the combination model.

German using CCM					
Overproposals		Underproposals		Crossing	
ADJA NN	461	APPR ART NN	97	ADJA NN	30
ART NN	430	APPR NN	84	ART NN VVPP	23
ART ADJA NN	94	APPR NE	46	CARD NN	18
KON NN	71	NE NE	32	NN VVPP	16
CARD NN	67	APPR ADJA NN	31	NE NE	15
PPOSAT NN	37	ADV ADJD	24	VVPP VAINF	13
ADJA NN NE	36	ADV ADV	23	ART NN PTKVZ	12
APPRART NN	33	APPR ART ADJA NN	21	NE NN	12
NE NE	30	NN NE	20	NE VVPP	12
ART NN VVPP	29	NN KON NN	19	ART NN VVINF	11

German using CCM+DMV					
Overproposals		Underproposals		Crossing	
ADJA NN	461	NE NE	30	ADJA NN	30
ART NN	430	NN KON NN	22	CARD NN	18
ART ADJA NN	94	NN NE	12	NE NE	17
KON NN	71	APPR NE	9	APPR NN	15
APPR NN	68	ADV PTKNEG	9	ART NN ART NN	14
CARD NN	67	VVPP VAINF	9	NE ADJA NN	11
NE ADJA NN	62	ADV ADJA	9	ADV ADJD	9
NE NE	38	ADV CARD	9	NN APPR NN	8
PPOSAT NN	37	ADJD ADJA	8	ADV NN	7
APPR ART NN	36	CARD CARD	7	APPRART NN NN	7

Figure 6.8: Sequences most frequently overproposed, underproposed, and proposed in locations crossing a gold bracket for German, for the CCM and the combination model.



(a) (b)

Figure 6.9: Change in parse quality as maximum sentence length increases: (a) CCM alone vs. combination and (b) DMV alone vs. combination.

the combination degrades substantially more slowly with sentence length than the CCM alone. This is not too surprising: the CCM's strength is finding common short constituent chunks: the DMV's representation is less scale sensitive at inference time. What is a little surprising is that the DMV and the combination actually converge in dependency accuracy as sentences get longer – this may well be because as sentences get longer, the pressure from the CCM gets relatively weaker: it is essentially agnostic about longer spans.

6.4 Conclusion

We have presented a successful new dependency-based model for the unsupervised induction of syntactic structure, which picks up the key ideas that have made dependency models successful in supervised statistical parsing work. We proceeded to show that it works cross-linguistically. We then demonstrated how this model could be combined with the constituent-induction model of chapter 5 to produce a combination which, in general, substantially outperforms either individual model, on either metric. A key reason that these models are capable of recovering structure more accurately than previous work is that they minimize the amount of hidden structure that must be induced. In particular, neither model attempts to learn intermediate, recursive categories with no direct connection to surface statistics. For example, the CCM models nesting but not recursion. The dependency model is a recursive model, but each tree node is headed by one of the leaves it dominates, so no hidden categories must be posited. Moreover, in their basic forms presented here, neither

model (nor their combination) requires any symmetry-breaking perturbations at initialization.

Chapter 7

Conclusions

There is a great deal of structure in human languages that is not explicit in the observed input. One way or another, human language learners figure out how to analyze, process, generalize, and produce languages to which they are exposed. If we wish to build systems which interact with natural languages, we either have to take a supervised approach and supply detailed annotation which makes all this hidden structure explicit, or we have to develop methods of inducing this structure automatically. The former problem is well-studied and well-understood; the latter problem is merely well-studied. This work has investigated a specific corner of the language learning task: inducing constituency and dependency tree structured analyses given only observations of grammatical sentences (or word-class sequences). We have demonstrated for this task several systems which exceed previous systems' performance in extracting linguistically reasonable structure. Hopefully, we have also provided some useful ideas about what does and does not work for this task, both in our own systems and in other researchers' work.

Many open questions and avenues of research remain, ranging from the extremely technical to the extremely broad. On the narrow side, machine learning issues exist for the present models. Aside from the multi-class CCM, the models have the virtue that symbol symmetries do not arise, so randomness is not needed at initialization. However, all models are sensitive to initialization to some degree. Indeed, for the CCM, one of the contributions of this work is the presentation of a better uniform initializer. In addition, the CCM model family is probabilistically mass deficient; it redundantly generates the observations, and,

most severely, its success may well rely on biases latent in this redundant generation. It performs better on corpora with shorter sentences than longer sentences; part of this issue is certainly that the linear sequences get extremely sparse for long sentences.

A little more broadly, the DMV models, which are motivated by dependency approaches that have performed well in the supervised case, still underperform the theoretically less satisfying CCM models. Despite an enduring feeling that lexical information beyond word-class must be useful for learning language, it seems to be a statistical distraction in some cases. General methods for starting with low-capacity models and gradually releasing model parameters are needed – otherwise we will be stuck with a trade-off between expressivity and learnability that will cap the achievable quality of inducible models.

Much more broadly, an ideal language learning system should not be disconnected from other aspects of language understanding and use, such as the context in which the utterances are situated. Without attempting to learn the meaning of sentences, success at learning their grammatical structure is at best an illuminating stepping stone to other tasks and at worst a data point for linguists interested in nativism. Moreover, from the standpoint of an NLP researcher in need of a parser for a language lacking supervised tools, approaches which are weakly supervised, requiring, say 100 or fewer example parses, are likely to be just as reasonable as fully unsupervised methods, and one could reasonably hope that they would provide better results. In fact, from an engineering standpoint, supplying a few parses is generally much easier than tuning an unsupervised algorithm for a specific language.

Nonetheless, it is important to emphasize that this work has shown that much progress in the unsupervised learning of real, broad-coverage parsers can come from careful understanding of the interaction between the representation of a probabilistic model and what kinds of trends it detects in the data. We can not expect that unsupervised methods will ever exceed supervised methods in cases where there is plenty of labeled training data, but we can hope that, when only unlabeled data is available, unsupervised methods will be important, useful tools, which additionally can shed light on how human languages are structured, used, and learned.

Appendix A

Calculating Expectations for the Models

A.1 Expectations for the CCM

In estimating parameters for the CCM model, the computational bottleneck is the E-step, where we must calculate posterior expectations of various tree configurations according to a fixed parameter vector (chapter 5). This section gives a cubic dynamic program for efficiently collecting these expectations.

The CCM model family is parameterized by two kinds of multinomials: the class-conditional span generation terms $\mathrm{P}_{\mathrm{SPAN}}(\alpha|c)$ and the class-conditional context generation terms $\mathrm{P}_{\mathrm{CONTEXT}}(\beta|c)$, where c is a boolean indicating the constituency of the span, α is the sequence filling that span, and β is the local linear context of that span. The score assigned to a sentence $s_{0,n}$ under a single bracketing B is

$$\mathrm{P}(s, B) = \mathrm{P}_{\mathrm{TREE}}(B) \prod_{\langle i,j \rangle} \mathrm{P}_{\mathrm{SPAN}}(\alpha(i,j,s)|B_{ij}) \mathrm{P}_{\mathrm{CONTEXT}}(\beta(i,j,s)|B_{ij})$$

In $\mathrm{P}_{\mathrm{TREE}}(B)$, the bracketings B with non-zero mass are in one-to-one correspondence with the set of binary tree structures T. Therefore, we can rewrite this expression in terms of the

constituent brackets in the tree $T(B)$.

$$\mathrm{P}(s,B) = \mathrm{P}_{\mathrm{TREE}}(B) \prod_{\langle i,j\rangle \in T(B)} \mathrm{P}_{\mathrm{SPAN}}(\alpha(i,j,s)|\textit{true})\mathrm{P}_{\mathrm{CONTEXT}}(\beta(i,j,s)|\textit{true}) \prod_{\langle i,j\rangle \notin T(B)} \mathrm{P}_{\mathrm{SPAN}}(\alpha(i,j,s)|\textit{false})\mathrm{P}_{\mathrm{CONTEXT}}(\beta(i,j,s)|\textit{false})$$

Since most spans in any given tree are distituents, we can also calculate the score for a bracketing B by starting with the score for the all-distituent bracketing and multiplying in correction factors for the spans which do occur as constituents in B:

$$\mathrm{P}(s,B) = \mathrm{P}_{\mathrm{TREE}}(B) \prod_{\langle i,j\rangle} \mathrm{P}_{\mathrm{SPAN}}(\alpha(i,j,s)|\textit{false})\mathrm{P}_{\mathrm{CONTEXT}}(\beta(i,j,s)|\textit{false}) \prod_{\langle i,j\rangle \in T(B)} \frac{\mathrm{P}_{\mathrm{SPAN}}(\alpha(i,j,s)|\textit{false})\mathrm{P}_{\mathrm{CONTEXT}}(\beta(i,j,s)|\textit{false})}{\mathrm{P}_{\mathrm{SPAN}}(\alpha(i,j,s)|\textit{true})\mathrm{P}_{\mathrm{CONTEXT}}(\beta(i,j,s)|\textit{true})}$$

Since all binary trees have the same score in $\mathrm{P}_{\mathrm{TREE}}$, and the all-distituent product does not depend on the bracketing, we can write this as

$$\mathrm{P}(s,B) = K(s) \prod_{\langle i,j\rangle \in T(B)} \phi(i,j,s)$$

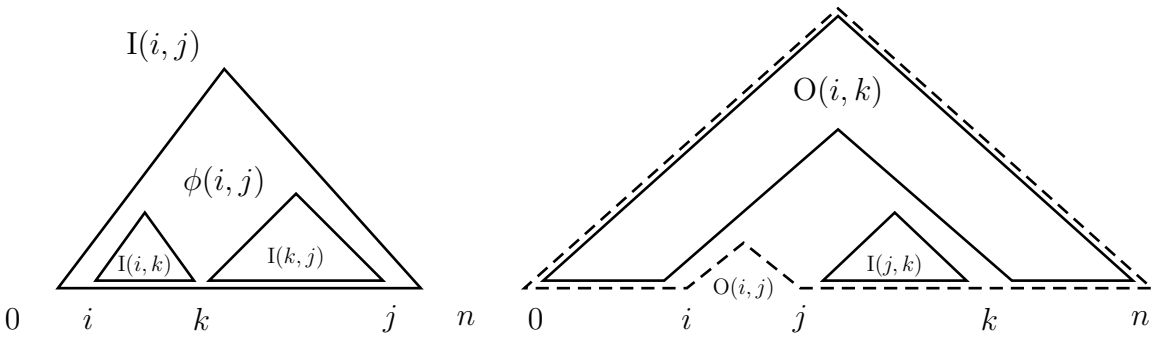
where

$$\phi(i,j,s) = \frac{\mathrm{P}_{\mathrm{SPAN}}(\alpha(i,j,s)|\textit{true})\mathrm{P}_{\mathrm{CONTEXT}}(\beta(i,j,s)|\textit{true})}{\mathrm{P}_{\mathrm{SPAN}}(\alpha(i,j,s)|\textit{false})\mathrm{P}_{\mathrm{CONTEXT}}(\beta(i,j,s)|\textit{false})}$$

and $K(s)$ is some sentence-specific constant. This expression for $\mathrm{P}(s,B)$ now factors according to the nested tree structure of $T(B)$. Therefore, we can define recursions analogous to the standard inside/outside recurrences and use these to calculate the expectations we're interested in.

First, we define $\mathrm{I}(i,j,s)$, which is analogous to the inside score in the inside-outside algorithm for PCFGs.

$$\mathrm{I}(i,j,s) = \sum_{T \in \mathcal{T}(j-i)} \prod_{\langle a,b\rangle : \langle a-i, b-i\rangle \in T} \phi(s,a,b)$$



(a) Inside recurrence (b) Outside recurrence

Figure A.1: The inside and outside configurational recurrences for the CCM.

In other words, this is the sum, over all binary tree structures T spanning $\langle i, j\rangle$, of the products of the local ϕ scores of the brackets in those trees (see figure A.1). This quantity has a nice recursive decomposition:

$$\mathrm{I}(i,j,s) = \begin{cases} \phi(i,j,s) \sum_{i<k<j} \mathrm{I}(i,k,s)\mathrm{I}(k,j,s) & \text{if } j-i>1 \\ \phi(i,j,s) & \text{if } j-i=1 \\ 0 & \text{if } j-i=0 \end{cases}$$

From this recurrence, either a dynamic programming solution or a memoized solution for calculating the table of I scores in time $O(n^3)$ and space $O(n^2)$ is straightforward.

Similarly, we define $\mathrm{O}(i,j,s)$, which is analogous to the outside score for PCFGs:

$$\mathrm{O}(i,j,s) = \sum_{T\in\mathcal{T}(n-(j-i-1))} \prod_{\langle a,b\rangle\neq\langle i,j\rangle:\langle a,b-(j-i-1)\rangle\in T} \phi(a,b,s)$$

This quantity is the sum of the scores of all tree structures outside the $\langle i, j\rangle$ bracket (again see figure A.1). Note that here, O excludes the factor for the local score at $\langle i, j\rangle$. The

outside sum also decomposes recursively:

$$\mathrm{O}(i,j,s) = \begin{cases} \phi(i,j,s)\sum_{0\le k<i}\mathrm{I}(k,i,s)\mathrm{O}(k,j,s) + \sum_{j<k\le n}\mathrm{I}(j,k,s)\mathrm{O}(i,k,s) & \text{if } j-i<n \\ 1 & \text{if } j-i=n \end{cases}$$

Again, the table of O values can be computed by dynamic programming or memoization.

The expectations we need for reestimation of the CCM are the posterior bracket counts $\mathrm{P}_{\mathrm{BRACKET}}(i,j|s)$, the fraction of trees (bracketings) that contain the span $\langle i,j,\rangle$ as a constituent.

$$\mathrm{P}_{\mathrm{BRACKET}}(i,j|s) = \frac{\sum_{B:B(i,j)=\mathit{true}} P(s,B)}{\sum_B P(s,B)}$$

We can calculate the terms in this expression using the I and O quantities. Since the set of trees containing a certain bracket is exactly the cross product of partial trees inside that bracket and partial trees outside that bracket, we have

$$\sum_{B:B(i,j)=\mathit{true}} P(s,B) = K(s)\mathrm{I}(i,j,s)\mathrm{O}(i,j,s)$$

and

$$\sum_B P(s,B) = K(s)\mathrm{I}(0,n,s)\mathrm{O}(0,n,s)$$

Since the constants $K(s)$ cancel and $\mathrm{O}(0,n,s)=1$, we have

$$\mathrm{P}_{\mathrm{BRACKET}}(i,j|s) = \frac{\mathrm{I}(i,j,s)\mathrm{O}(i,j,s)}{\mathrm{I}(0,n,s)}$$

A.2 Expectations for the DMV

The DMV model can be most simply (though not most efficiently) described as decomposing over a lexicalized tree structure, as shown in figure A.2. These trees are essentially context-free trees in which the terminal symbols are $w \in W$ for some terminal vocabulary W (here, W is the set of word-classes). The non-terminal symbols are $\overrightarrow{w}$, $\overleftarrow{w}$, $\overleftrightarrow{w}$, $\overrightarrow{\overrightarrow{w}}$, and $\overline{w}$, for $w \in W \cup \{\blacklozenge\}$. The productions are of the following forms:

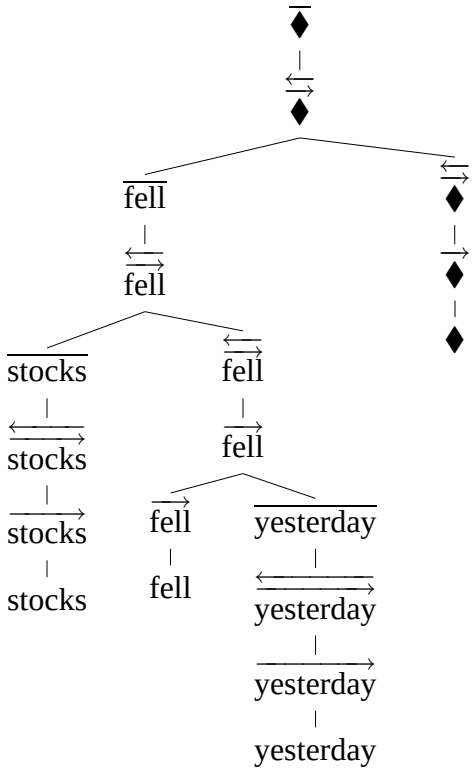


Figure A.2: A lexicalized tree in the fully articulated DMV model.

Head choice (right-first)	$\overrightarrow{w} \rightarrow w$
Head choice (left-first)	$\overleftarrow{w} \rightarrow w$
Right-first right attachment	$\overrightarrow{h} \rightarrow \overrightarrow{h} \ \ \overline{a}$
Right-first right stop	$\overleftrightarrow{h} \rightarrow \overrightarrow{h}$
Right-first left attachment	$\overleftrightarrow{h} \rightarrow \overline{a} \ \ \overleftrightarrow{h}$
Right-first seal	$\overline{h} \rightarrow \overleftrightarrow{h}$
Left-first left attachment	$\overleftarrow{h} \rightarrow \overline{a} \ \ \overleftarrow{h}$
Left-first left stop	$\overrightleftarrow{h} \rightarrow \overleftarrow{h}$
Left-first right attachment	$\overrightleftarrow{h} \rightarrow \overrightleftarrow{h} \ \ \overline{a}$
Left-first seal	$\overline{h} \rightarrow \overrightleftarrow{h}$

We imagine all sentences to end with the symbol ♦, and treat $\overline{\blacklozenge}$ as the start symbol. Trees with root $\overline{h}$ are called *sealed* trees, since their head h cannot take any further arguments in this grammar topology. Trees with roots $\overleftrightarrow{h}$ and $\overrightleftarrow{h}$ are called *half-sealed* trees, since they can only take arguments to the final attachment side (left and right, respectively). Trees rooted at $\overrightarrow{h}$ and $\overleftarrow{h}$ are called *unsealed* since they can take arguments to either side (eventually).

Derivations of a sentence in the DMV dependency model correspond to parses with the above grammar. However, each configuration is scored according to a head-outward derivation (Collins 1999), rather than a top-down rewrite. For example, the right-first head choice is thought of as deciding, given the head, that in this derivation the head will attach its right arguments before its left arguments. This configuration incurs the probability factor $\mathrm{P}_{\mathrm{ORDER}}(\textit{right-first}|w)$. The other configurations indicate attachments or stopping probabilities, as described in section 6.2.

We define inside and outside recurrences over items $(\mathrm{X} : i, j)$ in the standard way. The base cases are projections of terminals, which represent the point where we decide whether a word w will take its arguments to the right then left, or left then right, X-bar style.

$$\mathrm{P}_{\mathrm{INSIDE}}(x, i, i+1) = \begin{cases} \mathrm{P}_{\mathrm{ORDER}}(\textit{left-first}|w) & \text{if } x = \overleftarrow{w} \\ \mathrm{P}_{\mathrm{ORDER}}(\textit{right-first}|w) & \text{if } x = \overrightarrow{w} \end{cases}$$

For spans greater than one, we have to sum over the various decompositions. Sealed scores are expressed in terms of half-sealed scores:

$$\begin{aligned}\mathrm{P}_{\text{INSIDE}}(\overline{w}, i, j) = \mathrm{P}_{\text{STOP}}(\text{stop}|w, \text{left}, \text{adj}(i, w))\mathrm{P}_{\text{INSIDE}}(\overleftrightarrow{\overrightarrow{w}}, i, j) + \\ \mathrm{P}_{\text{STOP}}(\text{stop}|w, \text{right}, \text{adj}(j, w))\mathrm{P}_{\text{INSIDE}}(\overleftrightarrow{\text{[}} w, i, j)\end{aligned}$$

Half-sealed scores are expressed in terms of either smaller half-sealed scores or unsealed scores:

$$\begin{aligned}\mathrm{P}_{\text{INSIDE}}(\overleftarrow{\overrightarrow{w}}, i, j) = \Bigg(\sum_k \sum_a \mathrm{P}_{\text{STOP}}(\neg\text{stop}|w, \text{left}, \text{adj}(k, w))\mathrm{P}_{\text{ATTACH}}(a|w, \text{left}) \times \\ \mathrm{P}_{\text{INSIDE}}(\overline{a}, i, k)\mathrm{P}_{\text{INSIDE}}(\overleftarrow{\overrightarrow{w}}, k, j)\Bigg) + \\ \mathrm{P}_{\text{STOP}}(\text{stop}|w, \text{right}, \text{adj}(j, w))\mathrm{P}_{\text{INSIDE}}(\overrightarrow{w}, i, j) \\ \mathrm{P}_{\text{INSIDE}}(\overleftrightarrow{\overrightarrow{w}}, i, j) = \Bigg(\sum_k \sum_a \mathrm{P}_{\text{STOP}}(\neg\text{stop}|w, \text{right}, \text{adj}(k, w))\mathrm{P}_{\text{ATTACH}}(a|w, \text{right}) \times \\ \mathrm{P}_{\text{INSIDE}}(\overleftrightarrow{\overrightarrow{w}}, i, k)\mathrm{P}_{\text{INSIDE}}(\overline{a}, k, j)\Bigg) + \\ \mathrm{P}_{\text{STOP}}(\text{stop}|w, \text{left}, \text{adj}(i, w))\mathrm{P}_{\text{INSIDE}}(\overleftarrow{w}, i, j)\end{aligned}$$

Note the dependency on adjacency: the function $\text{adj}(i, w)$ indicates whether the index i is adjacent to word w (on either side).[1] Unsealed scores (for spans larger than one) are

[1] Note also that we're abusing notation so that w indicates not just a terminal symbol, but a specific instance of that symbol in the sentence.

expressed in terms of smaller unsealed scores:

$$\mathrm{P}_{\mathrm{INSIDE}}(\overrightarrow{w}, i, j) = \sum_k \sum_a \mathrm{P}_{\mathrm{STOP}}(\neg\mathrm{stop}|w, \mathrm{right}, \mathrm{adj}(k, w))\mathrm{P}_{\mathrm{ATTACH}}(a|w, \mathrm{right}) \times \mathrm{P}_{\mathrm{INSIDE}}(\overrightarrow{w}, i, k)\mathrm{P}_{\mathrm{INSIDE}}(\overline{a}, k, j)$$

$$\mathrm{P}_{\mathrm{INSIDE}}(\overleftarrow{w}, i, j) = \sum_k \sum_a \mathrm{P}_{\mathrm{STOP}}(\neg\mathrm{stop}|w, \mathrm{left}, \mathrm{adj}(k, w))\mathrm{P}_{\mathrm{ATTACH}}(a|w, \mathrm{left}) \times \mathrm{P}_{\mathrm{INSIDE}}(\overline{a}, i, k)\mathrm{P}_{\mathrm{INSIDE}}(\overrightarrow{w}, k, j)$$

The outside recurrences for $\mathrm{P}_{\mathrm{OUTSIDE}}(x, i, j)$ are similar. Both can be calculated in $O(n^5)$ time using the standard inside-outside algorithms for headed (lexicalized) PCFGs or memoization techniques. Of course, the $O(n^4)$ and $O(n^3)$ techniques of Eisner and Satta (1999) apply here, as well, but not in the case of combination with the CCM model, and are slightly more complex to present.

In any case, once we have the inside and outside scores, we can easily calculate the fraction of trees over a given sentence which contain any of the structural configurations which are necessary to re-estimate the model multinomials.

A.3 Expectations for the CCM+DMV Combination

For the combination of the CCM and DMV models, we used the simplest technique which admitted a dynamic programming solution. For any lexicalized derivation tree structure of the form shown in figure A.2, we can read off a list of DMV derivation stops (stops, attachments, etc.). However, we can equally well read off a list of assertions that certain sequences and their contexts are constituents or distituents. Both models therefore assign a score to a lexicalized derivation tree, though multiple distinct derivation trees will contain the same set of constituents.

To combine the models, we took lexicalized derivation trees T and scored them with

$$\mathrm{P}_{\mathrm{COMBO}}(T) = \mathrm{P}_{\mathrm{CCM}}(T)\mathrm{P}_{\mathrm{DMV}}(T)$$

The quantity $\mathrm{P}_{\mathrm{COMBO}}$ is a deficient probability function, in that, even if $\mathrm{P}_{\mathrm{CCM}}$ were not itself deficient, the sum of probabilities over all trees T will be less than one.[2]

Calculating expectations with respect to $\mathrm{P}_{\mathrm{COMBO}}$ is almost exactly the same as working with $\mathrm{P}_{\mathrm{DMV}}$. We use a set of $O(n^5)$ recurrences, as in section A.2. The only difference is that we must premultiply all our probabilities by the CCM base product

$$\prod_{\langle i,j \rangle} \mathrm{P}_{\mathrm{SPAN}}(\alpha(i,j,s)|\textit{false})\mathrm{P}_{\mathrm{CONTEXT}}(\beta(i,j,s)|\textit{false})$$

and we must multiply in the local CCM correction factor $\phi(i,j,s)$ at each constituent that spans more than one terminal. To do the latter, we simply adjust the binary-branching recurrences above. Instead of:

$$\begin{aligned}
\mathrm{P}_{\mathrm{INSIDE}}(\overleftarrow{\overrightarrow{w}}, i, j) = \Bigg(\sum_k \sum_a \mathrm{P}_{\mathrm{STOP}}(\neg\mathrm{stop}|w, \mathrm{left}, \mathrm{adj}(k,w))\mathrm{P}_{\mathrm{ATTACH}}(a|w,\mathrm{left}) \times \\
\mathrm{P}_{\mathrm{INSIDE}}(\overline{a}, i, k)\mathrm{P}_{\mathrm{INSIDE}}(\overleftarrow{\overrightarrow{w}}, k, j) \Bigg) + \\
\mathrm{P}_{\mathrm{STOP}}(\mathrm{stop}|w, \mathrm{right}, \mathrm{adj}(j,w))\mathrm{P}_{\mathrm{INSIDE}}(\overrightarrow{w}, i, j)
\end{aligned}$$

$$\begin{aligned}
\mathrm{P}_{\mathrm{INSIDE}}(\overleftarrow{\overrightarrow{w}}, i, j) = \Bigg(\sum_k \sum_a \mathrm{P}_{\mathrm{STOP}}(\neg\mathrm{stop}|w, \mathrm{right}, \mathrm{adj}(k,w))\mathrm{P}_{\mathrm{ATTACH}}(a|w,\mathrm{right}) \times \\
\mathrm{P}_{\mathrm{INSIDE}}(\overleftarrow{\overrightarrow{w}}, i, k)\mathrm{P}_{\mathrm{INSIDE}}(\overline{a}, k, j) \Bigg) + \\
\mathrm{P}_{\mathrm{STOP}}(\mathrm{stop}|w, \mathrm{left}, \mathrm{adj}(i,w))\mathrm{P}_{\mathrm{INSIDE}}(\overleftarrow{w}, i, j)
\end{aligned}$$

[2]Except in degenerate cases, such as if both components put mass one on a single tree.

we get

$$\mathrm{I}(\overleftrightarrow{w}, i, j|s) = \Bigg(\sum_k \sum_a \mathrm{P}_{\mathrm{STOP}}(\neg\mathrm{stop}|w, \mathrm{left}, \mathrm{adj}(k, w))\mathrm{P}_{\mathrm{ATTACH}}(a|w, \mathrm{left})\mathrm{P}_{\mathrm{INSIDE}}(\overline{a}, i, k) \times \mathrm{P}_{\mathrm{INSIDE}}(\overleftrightarrow{w}, k, j)\phi(i, j, s)\Bigg) + \mathrm{P}_{\mathrm{STOP}}(\mathrm{stop}|w, \mathrm{right}, \mathrm{adj}(j, w))\mathrm{P}_{\mathrm{INSIDE}}(\overrightarrow{w}, i, j)$$

$$\mathrm{I}(\overleftrightarrow{w}, i, j|s) = \Bigg(\sum_k \sum_a \mathrm{P}_{\mathrm{STOP}}(\neg\mathrm{stop}|w, \mathrm{right}, \mathrm{adj}(k, w))\mathrm{P}_{\mathrm{ATTACH}}(a|w, \mathrm{right}) \times \mathrm{P}_{\mathrm{INSIDE}}(\overleftrightarrow{w}, i, k)\mathrm{P}_{\mathrm{INSIDE}}(\overline{a}, k, j)\phi(i, j, s)\Bigg) + \mathrm{P}_{\mathrm{STOP}}(\mathrm{stop}|w, \mathrm{left}, \mathrm{adj}(i, w))\mathrm{P}_{\mathrm{INSIDE}}(\overleftarrow{w}, i, j)$$

and similarly

$$\mathrm{P}_{\mathrm{INSIDE}}(\overrightarrow{w}, i, j) = \sum_k \sum_a \mathrm{P}_{\mathrm{STOP}}(\neg\mathrm{stop}|w, \mathrm{right}, \mathrm{adj}(k, w))\mathrm{P}_{\mathrm{ATTACH}}(a|w, \mathrm{right}) \times \mathrm{P}_{\mathrm{INSIDE}}(\overrightarrow{w}, i, k)\mathrm{P}_{\mathrm{INSIDE}}(\overline{a}, k, j)$$

$$\mathrm{P}_{\mathrm{INSIDE}}(\overleftarrow{w}, i, j) = \sum_k \sum_a \mathrm{P}_{\mathrm{STOP}}(\neg\mathrm{stop}|w, \mathrm{left}, \mathrm{adj}(k, w))\mathrm{P}_{\mathrm{ATTACH}}(a|w, \mathrm{left}) \times \mathrm{P}_{\mathrm{INSIDE}}(\overline{a}, i, k)\mathrm{P}_{\mathrm{INSIDE}}(\overrightarrow{w}, k, j)$$

becomes

$$\mathrm{I}(\overrightarrow{w}, i, j|s) = \sum_k \sum_a \mathrm{P}_{\mathrm{STOP}}(\neg\mathrm{stop}|w, \mathrm{right}, \mathrm{adj}(k, w))\mathrm{P}_{\mathrm{ATTACH}}(a|w, \mathrm{right}) \times \mathrm{P}_{\mathrm{INSIDE}}(\overrightarrow{w}, i, k)\mathrm{P}_{\mathrm{INSIDE}}(\overline{a}, k, j)\phi(i, j, s)$$

$$\mathrm{I}(\overleftarrow{w}, i, j|s) = \sum_k \sum_a \mathrm{P}_{\mathrm{STOP}}(\neg\mathrm{stop}|w, \mathrm{left}, \mathrm{adj}(k, w))\mathrm{P}_{\mathrm{ATTACH}}(a|w, \mathrm{left}) \times \mathrm{P}_{\mathrm{INSIDE}}(\overline{a}, i, k)\mathrm{P}_{\mathrm{INSIDE}}(\overrightarrow{w}, k, j)\phi(i, j, s)$$

Again, the outside expressions are similar. Notice that the scores which were inside probabilities in the DMV case are now only sum-of-products of scores, relativized to the current sentence, just as in the CCM case.

Appendix B

Proofs

B.1 Closed Form for the Tree-Uniform Distribution

The tree-uniform distribution, $\mathrm{P}_{\mathrm{TREE}}(T|n)$, is simply the uniform distribution over the set of binary trees spanning n leaves. The statistic needed from this distribution in this work is the posterior bracketing distribution, $\mathrm{P}(i,j|n)$, the fraction of trees, according to the tree distribution, which contain the bracket $\langle i,j\rangle$. In the case of the tree-uniform distribution, these posterior bracket counts can be calculated in (basically) closed form.

Since each tree has equal mass, we know that $\mathrm{P}(i,j|n)$ is simply the number of trees containing the $\langle i,j\rangle$ bracket divided by the total number $T(n)$ of trees over n terminals. The latter is well-known to be $C(n-1)$, the $(n-1)$st Catalan number:

$$T(n) = C(n-1) = (\binom{2n-2}{n-1})/n$$

So how many trees over n leaves contain a bracket $b = \langle i,j\rangle$? Each such tree can be described by the pairing of a tree over $j-i$ leaves, which describes what the tree looks like inside the bracket b, and another tree, over $n-(j-i-1)$ leaves, which describes what the tree looks like outside of the bracket b. Indeed, the choices of inside and outside trees will give rise to all and only the trees over n symbols containing b. Therefore, the number

of trees with b is $T(j-i)T(n-(j-i-1))$, and the final fraction is

$$\mathrm{P}_{\mathrm{BRACKET}}(i,j|n) = \frac{T(j-i)T(n-(j-i-1))}{T(n)}$$

B.2 Closed Form for the Split-Uniform Distribution

The split-uniform distribution over the set of binary trees spanning n leaves, $\mathrm{P}_{\mathrm{SPLIT}}(T|n)$ is defined (recursively) as follows. If n is 1, there is only the trivial tree, T_1, which has conditional probability 1:

$$\mathrm{P}_{\mathrm{SPLIT}}(T_1|1) \equiv 1$$

Otherwise, there are $n-1$ options for top-level split points. One is chosen uniformly at random. This splits the n leaves into a set of k left leaves and $n-k$ right leaves, for some k. The left and right leaves are independently chosen from $\mathrm{P}_{\mathrm{SPLIT}}(\cdot|k)$ and $\mathrm{P}_{\mathrm{SPLIT}}(\cdot|n-k)$. Formally, for a specific tree T over n leaves consisting of a left child T_L over k leaves and a right child T_R over $n-k$ leaves:

$$\mathrm{P}_{\mathrm{SPLIT}}(T|n) \equiv \mathrm{P}(k|n)\mathrm{P}_{\mathrm{SPLIT}}(T_L|k)\mathrm{P}_{\mathrm{SPLIT}}(T_R|n-k)$$

where

$$\mathrm{P}(k|n) \equiv \frac{1}{n-1}$$

The statistics of this distribution needed in this work are the posterior bracket expectations $\mathrm{P}(i,j|n)$, the fraction of trees over n nodes which contain a given bracket $\langle i,j\rangle$ according to this distribution. This quantity can be recursively defined, as well. Since all trees contain a bracket over the entire sentence,

$$\mathrm{P}_{\mathrm{BRACKET}}(0,n|n) \equiv 1$$

For smaller spans, consider a tree T chosen a random from $\mathrm{P}_{\mathrm{SPLIT}}(\cdot|n)$. If it contains a bracket $b = \langle i,j\rangle$, then the bracket c immediately dominating b must be of the form $c = \langle i',j'\rangle$, where either $i' = i$ and $j < j'$ or $i' < i$ and $j = j'$. So how likely is it that T has bracket b? It should be clear that T will contain b if and only if it contains such a c,

and that c has the right immediate split point to produce b. We know from the top-down definition of $\mathrm{P}_{\mathrm{SPLIT}}(\cdot|n)$ that the location of the split point inside a bracket c is independent of of the chances of having built that c. Therefore, the total probability of a bracket b can be written (recursively) as:

$$\begin{aligned}\mathrm{P}_{\mathrm{BRACKET}}(i,j|n) &= \sum_{0\le i'<i} \mathrm{P}_{\mathrm{BRACKET}}(i',j|n)\mathrm{P}(i-i'|j-i') + \\ &\quad \sum_{j<j'\le n} \mathrm{P}_{\mathrm{BRACKET}}(i,j'|n)\mathrm{P}(j'-j|j'-i)\end{aligned}$$

The relevant solution to this recurrence is:

$$\mathrm{P}_{\mathrm{BRACKET}}(i,j|n) = \begin{cases} 1 & \text{if } i=0 \wedge j=n \\ 1/(j-i) & \text{if } i=0 \oplus j=n \\ 2/[(j-i)(j-i+1) & \text{if } 0<i<j<n \end{cases}$$

This solution was suggested by Noah Smith, *p.c.*, and can be proven by induction as follows. The base case ($i=0, j=n$) holds because all binary trees have a root bracket over all leaves. Now, assume for some k, all brackets of size $|j-i|>k$ obey the given solution. Consider $b=\langle i,j\rangle, |j-i|=k$.

Case I: Assume $i=0$. Then, we can express $\mathrm{P}_{\mathrm{BRACKET}}(i,j|n)$ in terms of larger brackets' likelihoods as

$$\begin{aligned}\mathrm{P}_{\mathrm{BRACKET}}(i,j|n) &= \sum_{j':j<j'\le n} \mathrm{P}_{\mathrm{BRACKET}}(0,j'|n)\mathrm{P}(j-0|j'-0) \\ &= \left[\sum_{j':j<j'<n} \frac{1}{j'}\frac{1}{j'-1}\right] + 1\frac{1}{n-1}\end{aligned}$$

A partial fraction expansion gives

$$\frac{1}{x}\frac{1}{x-1} = \frac{1}{x-1} - \frac{1}{x}$$

and so the sum telescopes:

$$\begin{aligned}
\mathrm{P}_{\mathrm{BRACKET}}(i,j|n) &= \left[\sum_{j':j<j'\leq n} \frac{1}{j'}\frac{1}{j'-1}\right] + \frac{1}{n-1} \\
&= \left[(\frac{1}{j-1}-\frac{1}{j}) + (\frac{1}{j}-\frac{1}{j+1}) + \ldots + (\frac{1}{n-2}-\frac{1}{n-1})\right] + \frac{1}{n-1} \\
&= \frac{1}{j-1}
\end{aligned}$$

and so the hypothesis holds.

Case II: Assume $j = n$. Since the definition of $\mathrm{P}_{\mathrm{SPLIT}}$ is symmetric, a similar argument holds as in case I.

Case III: Assume $0 < i < j < n$. Again we can expand $\mathrm{P}(i,j|n)$ in terms of known quantities:

$$\begin{aligned}
\mathrm{P}_{\mathrm{BRACKET}}(i,j|n) &= \sum_{j':j<j'\leq n} \mathrm{P}_{\mathrm{BRACKET}}(i,j'|n)\mathrm{P}(j-i|j'-i) + \\
&\quad \sum_{i':0\leq i'<i} \mathrm{P}_{\mathrm{BRACKET}}(i',j|n)\mathrm{P}(j-i|j-i')
\end{aligned}$$

It turns out that the two sums are equal:

$$S1(i,j|n) = \sum_{j':j<j'\leq n} \mathrm{P}_{\mathrm{BRACKET}}(i,j'|n)\mathrm{P}(j-i|j'-i) = \frac{1}{(j-i)(j-i+1)}$$

and

$$S2(i,j|n) = \sum_{i':0\leq i'<i} \mathrm{P}_{\mathrm{BRACKET}}(i',j|n)\mathrm{P}(j-i|j-i') = \frac{1}{(j-i)(j-i+1)}$$

We will show the $S1$ equality; the $S2$ equality is similar. Substituting the uniform split

probabilities, we get

$$S1(i,j|n) = \sum_{j':j<j'\leq n} \mathrm{P}_{\mathrm{BRACKET}}(i,j'|n)\frac{1}{j'-i-1}$$

and substituting the assumed values for the larger spans, we get

$$\begin{aligned} S1(i,j|n) &= \left[\sum_{j':j<j'<n} \mathrm{P}_{\mathrm{BRACKET}}(i,j'|n)\frac{1}{j'-i-1}\right] + \mathrm{P}_{\mathrm{BRACKET}}(i,n|n)\frac{1}{n-i-1} \\ &\quad \left[\sum_{j':j<j'<n} \frac{2}{(j'-i)(j'-i+1)}\frac{1}{j'-i-1}\right] + \frac{1}{n-i}\frac{1}{n-i-1} \end{aligned}$$

Here, the relevant partial fraction expansion is

$$\frac{2}{(x-1)(x)(x+1)} = \frac{1}{x-1} - \frac{2}{x} + \frac{1}{x+1}$$

Again the sum telescopes:

$$\begin{aligned} S1(i,j|n) &= \left[\sum_{j':j<j'<n} \frac{2}{(j'-i)(j'-i+1)}\frac{1}{j'-i-1}\right] + \frac{1}{n-i}\frac{1}{n-i-1} \\ &= \left[\left(\frac{1}{j-i-1} - \frac{2}{j-i} + \frac{1}{j-i+1}\right) + \left(\frac{1}{j-i} - \frac{2}{j-i+1} + \frac{1}{j-i+2}\right) + \cdots \right. \\ &\quad \left.\left(\frac{1}{n-i-2} - \frac{2}{n-i-1} + \frac{1}{n-i}\right)\right] + \left(\frac{1}{n-i-1} + \frac{1}{n-i}\right) \\ &= \left[\frac{1}{j-i-1} - \frac{1}{j-i} - \frac{1}{n-i-1} + \frac{1}{n-i}\right] + \left(\frac{1}{n-i-1} + \frac{1}{n-i}\right) \\ &= \frac{1}{j-i-1} - \frac{1}{j-i} \\ &= \frac{1}{(j-i-1)(j-i)} \end{aligned}$$

Since $S2$ is similar, we have

$$\begin{aligned} \mathrm{P}(i,j|n) &= S1(i,j|n) + S2(i,j|n) \\ &= 2S1(i,j|n) \\ &= \frac{2}{(j-i-1)(j-i)} \end{aligned}$$

and so the hypothesis holds again.

Bibliography

Abney, S. P. 1987. *The English Noun Phrase in its Sentential Aspect*. PhD thesis, MIT.

Adriaans, P., and E. Haas. 1999. Grammar induction as substructural inductive logic programming. In J. Cussens (Ed.), *Proceedings of the 1st Workshop on Learning Language in Logic*, 117–127, Bled, Slovenia.

Angluin, D. 1990. Negative results for equivalence theories. *Machine Learning* 5(2).

Baker, C. L., and J. J. McCarthy (Eds.). 1981. *The logical problem of language acquisition*. Cambridge, Mass.: MIT Press.

Baker, J. K. 1979. Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolf (Eds.), *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, 547–550.

Black, E., S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings, Speech and Natural Language Workshop*, 306–311, Pacific Grove, CA. DARPA.

Brill, E. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Meeting of the Association for Computational Linguistics*, 259–265.

Brown, R., and C. Hanlon. 1970. Derivational complexity and order of acquisition in child

speech. In J. R. Hayes (Ed.), *Cognition and the Development of Language*. New York: Wiley.

Carroll, G., and E. Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In C. Weir, S. Abney, R. Grishman, and R. Weischedel (Eds.), *Working Notes of the Workshop Statistically-Based NLP Techniques*, 1–13. Menlo Park, CA: AAAI Press.

Chang, N., and O. Gurevich. 2004. Context-driven construction learning. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*.

Charniak, E. 1996. Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI '96)*, 1031–1036.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 1)*, 132–139.

Charniak, E., K. Knight, and K. Yamada. 2003. Syntax-based language models for machine translation. In *Proceedings of MT Summit IX*.

Chelba, C., and F. Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Meeting of the Association for Computational Linguistics (ACL 36/COLING 17)*, 225–231.

Chen, S. F. 1995. Bayesian grammar induction for language modeling. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics (ACL 33)*, 228–235.

Chomsky, N. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.

Chomsky, N. 1986. *Knowledge of Language: Its Nature, Origin and Use*. New York: Praeger.

Clark, A. 2000. Inducing syntactic categories by context distribution clustering. In *The Fourth Conference on Natural Language Learning*.

Clark, A. 2001a. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *The Fifth Conference on Natural Language Learning*.

Clark, A. 2001b. *Unsupervised Language Acquisition: Theory and Practice*. PhD thesis, University of Sussex.

Clark, A. 2003. Combining distributional and morphological information in part of speech induction. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

Collins, M. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Demetras, M. J., K. N. Post, and C. E. Snow. 1986. Feedback to first language learners: the role of repetitions and clarification questions. *Journal of Child Language* 13:275–292.

Eisner, J. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING 16)*, 340–345.

Eisner, J., and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 457–464.

Finch, S., and N. Chater. 1992. Bootstrapping syntactic categories using statistical methods. In W. Daelemans and D. Powers (Eds.), *Background and Experiments in Machine Learning of Natural Language*, 229–235, Tilburg University. Institute for Language Technology and AI.

Finch, S. P. 1993. *Finding Structure in Language*. PhD thesis, University of Edinburgh.

Fodor, J. 1983. *Modularity of mind*. Cambridge, Mass.: MIT Press.

Gold, E. M. 1967. Language identification in the limit. *Information and Control* 10:447–474.

Halliday, M. A. K. 1994. *An introduction to functional grammar*. London: Edward Arnold. 2nd edition.

Hirsh-Pasek, K., R. Treiman, and M. Schneiderman. 1984. Brown and Hanlon revisited: mothers' sensitivity to ungrammatical forms. *Journal of Child Language* 11:81–88.

Hofmann, T. 1999. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence*, Stockholm.

Horning, J. J. 1969. *A study of grammatical inference*. PhD thesis, Stanford.

Jackendoff, R. 1996. *The architecture of the language faculty*. Cambridge, Mass.: MIT Press.

Kit, C. 1998. A goodness measure for phrase structure learning via compression with the mdl principle. In *Proceedings of the European Summer School in Logic, Language and Information Student Session*.

Klein, D., and C. D. Manning. 2001a. Distributional phrase structure induction. In *Proceedings of the Fifth Conference on Natural Language Learning (CoNLL 2001)*, 113–120.

Klein, D., and C. D. Manning. 2001b. Natural language grammar induction using a constituent-context model. In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14 (NIPS 2001)*, Vol. 1, 35–42. MIT Press.

Klein, D., and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL 40*, 128–135.

Klein, D., and C. D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In S. Becker, S. Thrun, and K. Obermayer (Eds.), *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.

Klein, D., and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 04)*.

Landauer, T. K., P. W. Foltz, and D. Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes* 25:259–284.

Langley, P., and S. Stromsten. 2000. Learning context-free grammars with a simplicity bias. In *Machine Learning: ECML 2000, 11th European Conference on Machine Learning, Barcelona, Catalonia, Spain, May 31 - June 2, 2000, Proceedings*, Vol. 1810, 220–228. Springer, Berlin.

Lari, K., and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* 4:35–56.

Magerman, D., and M. Marcus. 1990. Parsing a natural language using mutual information statistics. In *Proceedings of the Eighth National Conference on Artificial Intelligence*.

Manning, C. D., and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press.

Marcus, G. 1993. Negative evidence in language acquisition. *Cognition* 46(1):53–85.

Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics* 19:313–330.

Mel′čuk, I. A. 1988. *Dependency Syntax: theory and practice*. Albany, NY: State University of New York Press.

Merialdo, B. 1994. Tagging English text with a probabilistic model. *Computational Linguistics* 20(2):155–171.

Miller, P. H. 1999. *Strong Generative Capacity*. Stanford, CA: CSLI Publications.

Olivier, D. C. 1968. *Stochastic Grammars and Language Acquisition Mechanisms*. PhD thesis, Harvard University.

Paskin, M. A. 2002. Grammatical bigrams. In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Penner, S. 1986. Parental responses to grammatical and ungrammatical child utterances. *Child Development* 58:376–384.

Pereira, F., and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Meeting of the Association for Computational Linguistics (ACL 30)*, 128–135.

Pereira, F., N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL 31)*, 183–190.

Pinker, S. 1994. *The Language Instinct*. New York: William Morrow.

Pullum, G. K. 1996. Learnability, hyperlearning, and the poverty of the stimulus. In *Proceedings of the 22nd Annual Meeting of the Berkeley Linguistics Society*, Berkeley CA. Berkeley Linguistics Society.

Radford, A. 1988. *Transformational Grammar*. Cambridge: Cambridge University Press.

Roark, B. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics* 27:249–276.

Saffran, J. R., E. L. Newport, and R. N. Aslin. 1996. Word segmentation: the role of distributional cues. *Journal of Memory and Language* 35:606–621.

Schütze, H. 1993. Part-of-speech induction from scratch. In *The 31st Annual Meeting of the Association for Computational Linguistics (ACL 31)*, 251–258.

Schütze, H. 1995. Distributional part-of-speech tagging. In *Proceedings of the 7th Meeting of the European Chapter of the Association for Computational Linguistics (EACL 7)*, 141–148, San Francisco CA. Morgan Kaufmann.

Skut, W., T. Brants, B. Krenn, and H. Uszkoreit. 1998. A linguistically interpreted corpus of German newspaper texts. In *Proceedings of the European Summer School in Logic, Language and Information Workshop on Recent Advances in Corpus Annotations*.

Solan, Z., E. Ruppin, D. Horn, and S. Edelman. 2003. Automatic acquisition and efficient representation of syntactic structures. In S. Becker, S. Thrun, and K. Obermayer (Eds.), *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.

Stolcke, A., and S. M. Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In *Grammatical Inference and Applications: Proceedings of the Second International Colloquium on Grammatical Inference*. Springer Verlag.

van Zaanen, M. 2000. ABL: Alignment-based learning. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 18)*, 961–967.

Wolff, J. G. 1988. Learning syntax and meanings through optimization and distributional analysis. In Y. Levy, I. M. Schlesinger, and M. D. S. Braine (Eds.), *Categories and processes in language acquisition*, 179–215. Hillsdale, NJ: Lawrence Erlbaum.

Xue, N., F.-D. Chiou, and M. Palmer. 2002. Building a large-scale annotated Chinese corpus. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*.

Yuret, D. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. PhD thesis, MIT.