

ON THE PARAMETER SPACE OF GENERATIVE LEXICALIZED
STATISTICAL PARSING MODELS

Daniel M. Bikel

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2004

Mitchell P. Marcus
Supervisor of Dissertation

Benjamin C. Pierce
Graduate Group Chair

COPYRIGHT

Daniel M. Bikel

2004

For Granny

Acknowledgments

There are many people who have made this research possible, and whom I would like to thank. Before I ever reached Penn, I had a wonderful three years at BBN, where I first learned about statistical NLP from Scott Miller, Rich Schwartz and the ever-supportive Ralph Weischedel. For one of those years, I also learned a great deal from David Magerman, not only about statistical modeling, but also about hard-core programming. During my first two years at Penn, there was a large crop of graduate students who had a great impact on me: Jeff Reynar, Dan Melamed, Jason Eisner, Adwait Ratnaparkhi and Mike Collins. During several of my years as a graduate student, Anoop Sarkar and Dan Gildea were great sources of expertise and insight, and it has been a pleasure collaborating with my brilliant friend and colleague, David Chiang.

Of course, one's research can benefit from doing other things. Siome Klein Goldenstein, who was also a Ph.D. student in the CIS Department (and is now a Ph.D.), was, is and continues to be a great friend, who introduced me to, among other things, an eclectic and marvelous group of people who dance Argentine tango. I have immensely enjoyed laughing and dancing with April Marinell, Jackie Stahl, Paula Stein, Kelly Ray, Lesley Mitchell, Mariano Sana and Kristin Balmer. I also met Kathleen O'Banion, who has been a fantastic dance partner and friend. Just as I began to take up road bicycle racing, I met Kathleen's husband Frank Koniges. I have enjoyed his company literally for several thousand miles, including in the pre-Alpini and Dolimiti in Italy. I have also logged many memorable miles with my teammates from the Penn Cycling Team, so thanks go to my hearty racing friends Annalise Paaby, Sean Ott, Stefan Miltchev and especially to the uncompromising and indomitable Jeff Kralik.

Returning to the research itself, I would like to thank the brilliant professors of Penn who have

helped get me to this point, especially Mark Steedman, Lyle Ungar, Martha Palmer, Aravind Joshi and Fernando Pereira. It has also been a pleasure discussing many an issue with Professor Tony Kroch of the Linguistics Department.

I could not have asked for a better advisor than Mitch Marcus. Not only does he have great insight into many NLP problems, but he has always shown an unbridled optimism and enthusiasm for my research efforts, even when I myself felt otherwise. Mitch is a rather “hands off” advisor: he is there when you need him, but allows his students to explore their research areas, and worries about making sure they have all the resources they need to do it. This has been an ideal situation for me, and I am very grateful to Mitch for allowing me the freedom to think almost exclusively about my research. And when it comes to navigating the sometimes treacherous political waters of academic life, there is no better advisor and advocate than Mitch; for this I am also very grateful.

Finally, I’d like to thank Jenn Cappeto, who has shown great patience and understanding in the last two years of my doctoral studies, and I’d like to thank my family (Rita, Theo, Robbie and Judi, Bikels all), who, except for that one time, showed me nothing but love and support all these years.

ABSTRACT
ON THE PARAMETER SPACE OF GENERATIVE LEXICALIZED STATISTICAL PARSING
MODELS

Daniel M. Bikel

Supervisor: Mitchell P. Marcus

In this thesis, we apply as well as develop techniques and methodologies for the examination of the complex systems that are lexicalized statistical parsing models. The primary idea is that of treating the “model as data”, which is not a particular method, but a paradigm and a research methodology. Our argument is that lexicalized statistical parsing models have become increasingly complex, and therefore require thorough scrutiny, both to achieve the scientific aim of understanding what has been built thus far, and to achieve both the scientific *and* engineering goal of using that understanding for progress. In this thesis, we take a particular, dominant type of parsing model and perform a *macro analysis*, to reveal its core (and design a software engine that modularizes the periphery), and we also crucially perform a *detailed analysis*, which provides for the first time a window onto the efficacy of specific parameters. These analyses have not only yielded insight into the core model, but they have also enabled the identification of “inefficiencies” in our baseline model, such that those inefficiencies can be reduced to form a more compact model, or exploited for finding a better-estimated model with higher accuracy, or both.

Contents

Acknowledgments	v
1 Introduction	1
1.1 Parsing at a critical state	1
1.2 Motivation	3
1.3 Thesis	4
1.4 History-based models	6
1.5 Summary of this document	7
1.6 What this dissertation is not	8
2 Previous work	9
2.1 Seminal Work	10
2.2 Lexical relations	12
2.3 Stochastic lexicalized tree-adjoining grammars	13
2.4 Dependency parsing	14
2.5 The current era	14
2.6 Parser analysis	16
2.7 Recent work	17
2.8 Relation to the present work	20
3 Reproducing the baseline model: a complicated task	21
3.1 Motivation	22

3.2	Model Overview	23
3.3	Preprocessing training trees	24
3.3.1	Coordinated phrases	25
3.3.2	Pruning of unnecessary nodes	26
3.3.3	Adding base NP nodes (NPBs)	26
3.3.4	Repairing base NPs	27
3.3.5	Adding gap information	28
3.3.6	Relabeling subjectless sentences	28
3.3.7	Removing null elements	29
3.3.8	Raising punctuation	29
3.3.9	Identification of argument nonterminals	30
3.3.10	Stripping unused nonterminal augmentations	31
3.3.11	Repairing subjectless sentences	31
3.3.12	Head-finding	31
3.4	Training	33
3.4.1	Verb intervening	33
3.4.2	Skip certain trees	34
3.4.3	Unknown words	35
3.4.3.1	The threshold problem	35
3.4.3.2	Not handled in a uniform way	35
3.5	Parameter Classes and Their Estimation	36
3.5.1	The head parameter class	37
3.5.2	The subcat parameter class	37
3.5.3	The modifying nonterminal parameter class	38
3.5.4	The punctuation and coordinating conjunction parameter classes	39
3.5.4.1	Inconsistent model	39
3.5.4.2	History mechanism	42
3.5.5	The base NP model: a model unto itself	44
3.5.6	Parameter classes for priors on lexicalized nonterminals	45

3.5.7	Smoothing weights	45
3.5.7.1	Deficient model	46
3.5.7.2	Smoothing factors and smoothing terms	47
3.5.8	Modifier head-word generation	48
3.5.8.1	Smoothing and the last level of back-off	49
3.5.8.2	Unknown word mapping	49
3.5.9	The TOP parameter classes	52
3.6	Decoding	53
3.6.1	Chart item equivalence	53
3.6.2	Pruning	54
3.6.3	Unknown words and parts of speech	55
3.7	Evaluation: Effects of Unpublished Details	56
4	Designing a new parsing engine	59
4.1	Motivation	59
4.2	Background	60
4.2.1	Plug-'n'-play probability structures and sentence-level parallelism	60
4.2.2	Language independence	61
4.3	Design Overview	61
4.3.1	Language package	64
4.3.2	Probability structure objects	65
4.3.3	Flexible subcat frames	66
4.3.4	Flexible constraint-satisfaction system for constrained parsing	67
4.4	Efficiency	68
4.4.1	Probability-level parallelism	68
4.4.2	Optimizations	69
4.5	Summary	70

5	Parsing in multiple languages	71
5.1	Porting English parsers to Chinese	71
5.1.1	Models and modifications	71
5.1.2	Initial results	72
5.2	Current Chinese results	74
5.3	Discussion of Chinese results	77
5.4	Other languages	78
5.4.1	The Arabic Treebank	79
5.4.2	Classical Portuguese	80
5.4.3	Current projects	81
6	Independence assumptions, back-off and meta-parameters	83
6.1	Further discussion of the baseline model	83
6.1.1	Choice of heads	85
6.1.2	Lexical dependencies matter	86
6.2	Recasting a generative parsing model	88
6.3	Arguments and heads: using EM to reestimate the model	88
6.3.1	Motivation	89
6.3.2	Background	90
6.3.2.1	Head-lexicalization	90
6.3.2.2	Other tree transformations	91
6.3.2.3	Problems with heuristics	91
6.3.3	Rule-based augmentation	92
6.3.4	Unsupervised learning of augmentations	93
6.3.4.1	Background: tree adjoining grammar	93
6.3.4.2	Implementation	94
6.3.4.3	Experiment	97
6.4	True EM with a smoothed model	100
6.4.1	Experiments Redux	104

6.4.2	Random head-finding	106
6.4.3	Discussion of results	109
7	Model analysis: model as data	113
7.1	Motivation	114
7.2	Frequencies	115
7.2.1	Definitions and notation	115
7.2.2	Basic frequencies	116
7.2.3	Exploratory data analysis: a common distribution	117
7.3	Entropies	119
7.4	Distributional similarity and bilinguistic statistics	122
7.4.1	Distributional similarity	123
7.4.2	Results	124
7.5	Distributional Similarity and Parameter Selection	125
7.5.1	Metric and experiments	125
7.5.2	Related work	128
7.6	Discussion: a problem related to label bias	129
7.7	Chinese model analysis	130
7.7.1	Entropy analysis	130
7.7.2	Parameter selection	134
7.8	A closer look at the pruned parameters	135
8	Conclusion	141
8.1	Summary	141
8.2	Interpretability	142
8.3	Future work	143
8.3.1	Word classes	144
8.3.2	Parameter selection	147
8.3.3	Model analysis	148

A	Chinese head-finding heuristics	149
B	Complete list of parameter classes in Collins' Model 3	151

List of Tables

3.1	Back-off levels for P_{L_w}/P_{R_w} , the modifier head-word generation parameter classes	48
3.2	New parameter class for the generation of head words of modifying nonterminals	49
3.3	Back-off structure for $P_{TOP_{NT}}$ and P_{TOP_w}	53
3.4	Overall parsing results using only details found in (Collins, 1997; Collins, 1999)	56
3.5	Effects of independently removing/changing individual details on overall parsing performance	57
5.1	Initial results for BBN and TIG models on English and Chinese test sets.	73
5.2	Heuristics for marking arguments in the Chinese Treebank.	74
5.3	Sizes of training and test corpora for the three available versions of the Chinese Treebank.	76
5.4	Current results on the Chinese Treebank, versions 1.0, 3.0 and 4.0	76
5.5	Comparison of results on parsing the CTB3.0 test set using output of mxPOST versus gold-standard part-of-speech tags for unseen words.	78
5.6	Parsing performance on an early version of the Arabic Treebank, using a mapped version of the gold-standard part-of-speech tags.	80
6.1	Number of times our parsing engine was able to deliver a probability for the various levels of back-off of the mod-word generation model, P_{M_w} , when testing on Section 00	84
6.2	Results on §00 of the Penn Treebank with simplified head rules	86
6.3	Parsing performance with various models on §00 of the Penn Treebank	86

6.4	Results on §23 with the “unlexicalized” version of Collins’ Model 2 and the results of Klein and Manning (2003) and Magerman (1995).	88
6.5	Original EM results on blind test data	99
6.6	New EM results on blind test data	108
7.1	All eleven parameter classes in our emulation of Collins’ Model 2	116
7.2	Average counts and diversities of histories of the P_M and P_{M_w} parameter classes . .	117
7.3	Average entropies for each parameter class.	119
7.4	Entropy distribution statistics for P_M and P_{M_w}	120
7.5	Jensen-Shannon statistics for back-off parameters in P_{M_w}	125
7.6	Parsing results on Sections 00 and 23 with Collins’ Model 3, our emulation of Collins’ Model 2 and the reduced version at a threshold of 0.06	127
7.7	Percentage of parameters removed from each parameter class for the 0.06-reduced model	128
7.8	Comparison of two different head-finding rule sets on CTB3.0	134
7.9	CTB3.0 parsing results with the full model and the 0.06-reduced model.	135
7.10	Percentage of pruned parameters per class for both the Chinese and English models	135

List of Figures

3.1	An NP that constitutes a coordinated phrase.	26
3.2	A non-head NPB child of NP requires insertion of extra NP.	27
3.3	An NPB is “repaired”.	28
3.4	Raising punctuation: perverse case where multiple punctuation elements appear along a frontier of a subtree.	30
3.5	Head children are not exempt from being relabeled as arguments.	31
3.6	Head moves from right to left conjunct in a coordinated phrase, <i>except</i> when the parent nonterminal is NPB.	32
3.7	Example of the verb intervening feature	34
3.8	A fully lexicalized tree. The VP node is the head child of S.	37
3.9	A tree containing both punctuation and conjunction.	40
3.10	The Collins model assigns equal probability to these three trees.	41
3.11	Unknown word mapping scheme of Collins’ model	50
4.1	Pictorial design overview of new parsing engine.	62
6.1	Methodology for the development of a statistical parser.	90
6.2	A simple lexicalized parse tree.	91
6.3	Grammar and derivation for “John should leave tomorrow” with TAG.	94
6.4	Log-likelihood of the training data after each of the EM iterations using the full head-finding rule set for the initial model.	96
6.5	Original English EM results, starting with full rule set.	97

6.6	Original English EM results, starting with simplified rule set.	98
6.7	Original Chinese EM results, starting with full rule set.	99
6.8	Log-likelihood of the training data after each of the EM iterations using the full head-finding rule set for the initial model.	105
6.9	F-measure for all sentences of Section 00 with EM models, using full head-finding rule set for the initial model.	105
6.10	Log-likelihood of the training data after each of the EM iterations using the simplified head-finding rule set for the initial model.	106
6.11	F-measure for all sentences of Section 00 with EM models, using simplified head-finding rule set for the initial model.	107
6.12	Log-likelihood of the CTB1.0 training data after each of the EM iterations on the Chinese model.	107
6.13	F-measure for all sentences of the CTB1.0 development test set with EM models. .	108
6.14	Log-likelihood of reduced-size training set (Penn Treebank §§02–06) after each of the EM iterations, using a 5% random head-finder for the initial model.	110
6.15	F-measure for all sentences of the Penn Treebank development test set with EM models, using §§02–06 as training and a 5% random head-finder for the initial model.	110
7.1	A frequent P_{M_w} history context, illustrated as a tree fragment	117
7.2	Cumulative density function for the P_{M_w} history context illustrated in Figure 7.1. .	118
7.3	Probability versus word frequency for head words of NP-A(NN) in the PP construction.	119
7.4	Entropy distribution for the P_{M_w} parameters.	120
7.5	Total modifier word-generation entropy broken down by parent,head,modifier triple	121
7.6	Total modifier word-generation entropy for the CTB3.0 training corpus, broken down by parent,head,modifier triple	131
7.7	Chinese verbs associated with high-entropy P_{M_w} parameters	133
7.8	Percentage of total number of pruned P_{M_w} parameters, broken down by tag of head word, t_h	136

7.9	Percentage of parameters pruned from each subclass P_{M_w, t_h}	137
7.10	Percentage of total number of pruned P_{M_w} parameters in the CTB3.0-trained Chinese model, broken down by tag of head word, t_h	138
7.11	Percentage of parameters pruned from each subclass P_{M_w, t_h}	139
A.1	Chinese head-finding rules.	150

Chapter 1

Introduction

1.1 Parsing at a critical state

The Danish physicist Per Bak was a very polarizing figure, not only in his own field, but in every other field whose foundations he attempted to establish or shake. He is perhaps most well known for his theory of *self-organizing criticality* (Bak, 1996), which attempts to provide a coherent explanation for many types of complex systems: how they can be formed via very simple iterative processes, how they can self-organize and reach a critical state such that various fluctuations (or earthquakes, avalanches, cataclysms) of all sizes will happen according to a power law distribution. In the theory, the small fluctuations are part of the same process as the large ones, just at a different scale (the existence of a power law distribution directly implies a “scale free” phenomenon). Knowing the exponent of the power law distribution for a SOC system means we know the frequency with which different sized fluctuations occur, but the magnitude of the next fluctuation cannot be predicted. Since the fluctuations are due to the same process, a large “cataclysmic” fluctuation does not require some proportionally large, external event as its trigger, but merely for the system to be in its critical state. Another hallmark of self-organized critical systems is that they have “emergent properties”: properties develop at larger scales from the smaller-scale systems on which they are based. In the largest sense, chemistry somehow “emerges” from physics, geophysics emerges from astrophysics, biology emerges from chemistry and geophysics, and so on (Bak, 1996, p. 7). In

this way, life itself, Bak argues, may be a consequence of the SOC system that is the subject of much of physics, namely the quarks, gluons and other elementary particles that form the basis of all matter, and the “simple” rules that govern their interaction. Over eons and through only these small interactions, matter diverged into the biotic and non-biotic varieties we observe today, or so the theory would proffer.

It did not take long before this compelling theory caught the attention of software engineers. They noticed that software systems would often grow ever more complex with the addition of every new and perhaps small feature. At a certain point, they would develop “emergent properties” such that users would find usage patterns or capabilities not even intended by the designers. Also, generally at the same point at which these new properties emerged, the systems would appear to be in what Bak would call a critical state, such that the addition of any new feature or bug fix might trigger an “avalanche” of unintended—and often undesirable—consequences. In effect, the behavior of the system become unpredictable with respect to modification: sometimes a small modification would have little to no consequence, sometimes, but far less often, it would have drastic consequences.

Language has some of the hallmarks of a SOC system, in that it displays the tell-tale power law distribution in Zipf’s Law. A statistical parsing model that is based on supervised learning is inherently related to language, in that it is *induced* from the data on which it is trained. That is, a researcher designs the feature or parameter *schemata* to be included in a model, and upon coupling with training data (actual sentences and their parse trees), these schemata are instantiated as thousands—perhaps hundreds of thousands—of actual parameters or features. (We will expand on the subject of parameter schemata versus actual parameters in subsequent chapters.) After training and therefore instantiation of a particular model, the model can have a complexity that approaches the data on which it was trained.

Speaking informally, parsing models appear to be complex systems that are in a critical state. In the last fifteen years, the field of statistical parsing has had a rapid maturation with incredible progress, followed by a soft ceiling in performance, where it has been increasingly more difficult to improve on existing methods. Just as with so-called SOC systems, parsing models have seen

the addition of increasingly large numbers of well-understood simple mechanisms whose interactions can create complexities unintended and not understood by the designers of those simple mechanisms. One of the state-of-the-art parsing models of the last few years was built by Charniak (2000), who recently wrote that he is no longer working on his parser “because it has resisted all of my attempts to improve it” (Charniak, p.c.). His is not a unique situation.

On the other hand, it is entirely possible that the cause of this relative stagnation in parsing performance is due to the field getting “stuck” in the wrong paradigm. Even if this is true, it does not obviate the need to understand the current paradigm. At the very least, one would want this understanding when, say, moving to a radically new modeling technique, so as not to waste effort at modeling that which can already be shown to be unhelpful or even detrimental, and to help identify the kinds of features that are more promising.

1.2 Motivation

Parsing is the task of identifying one or more tree structures for a given sequence of words. One way to bound this search problem is to define a grammar *a priori* and only construct trees consistent with that grammar. Another way would be to construct a statistical model of $p(\mathbf{T}|\mathbf{S})$ based on observed data, where \mathbf{T} and \mathbf{S} are respectively variables over the space of trees and sentences, and then for each sentence search the space of trees for which the model assigns a non-zero probability. Such a model is called a *conditional model*. Still another, related way would be to construct a *generative* or *joint* model of trees and their words—notated $p(\mathbf{T}, \mathbf{S})$ —but then use it as a way to search through the conditional space. For our purposes here, the parsing problem is to find a single, most-likely tree structure for a given word sequence.

Indeed, the conditional models introduced by Magerman (1994) proved that high-performance parsing was possible—provided one has access to a Treebank (Marcus, Santorini, and Marcinkiewicz, 1993). However, it was the subsequent conditional and generative models of Collins (1996; 1997) that showed that a relatively straightforward decomposition of a lexicalized parse tree with very simple estimation methods could yield a significant increase in accuracy even over Magerman’s ground-breaking technique. The apparent simplicity of Collins’ generative 1997 model inspired

others to invent generative parsers, most notably Charniak (1997). Given the great success, then, of generative statistical parsing models over the last several years, it is perhaps remarkable how much we *don't* know about how they work, and why they work so well.

For example, it is easy to infer from the work of Collins (1997; 2000) and Charniak (1997) that the number of parameters in these generative models is proportional to $G \cdot |V|^2$, where G is a constant proportional to the size of the underlying, unlexicalized grammar and $|V|$ is the number of distinct lexical items. This number of parameters might make the most hardened statistician squeamish. But what is the exact number? What are the properties of the thousands or tens of thousands of distributions? More simply, what do those distributions look like? Which distributions are learned most or least well, and which are most or least informative? Does the model's power lie in the richness of the conditioning contexts of the estimated conditional probabilities, or does it lie somewhere in the back-off estimates, which use less information? Are there other dependencies that might be more beneficial than the ones already employed? Can we improve upon the heuristics that most head-driven models use to preprocess their training trees? Given the fact that these models contain very little that is language-dependent, would it be possible to extend them to other languages, and if so, would accuracy suffer? Can we characterize differences between a model for one language versus a model for another?

1.3 Thesis

In this thesis, we apply as well as develop techniques and methodologies for the examination of the complex systems that are lexicalized statistical parsing models. The primary idea is that of treating the “model as data”, which is not a particular method, but a paradigm and a research methodology. Our argument is that lexicalized statistical parsing models have become increasingly complex, and therefore require thorough scrutiny, both to achieve the scientific aim of understanding what has been built thus far, and to achieve both the scientific *and* engineering goal of using that understanding for progress. In this thesis, we take a particular, dominant type of parsing model and perform a *macro analysis*, to reveal its core (and design a software engine that modularizes the periphery), and we also crucially perform a *detailed analysis*, which provides for the first time a window onto the

efficacy of specific parameters. These analyses have not only yielded insight into the core model, but they have also “paid off”, in that they have enabled the identification of “inefficiencies” in our baseline model, such that those inefficiencies can be reduced to form a more compact model, or exploited for finding a better-estimated model with higher accuracy, or both.

In this thesis

- we revise or, in some cases, refute long-held assumptions in the NLP community relating to the use of bilexical parameters in the dominant class of lexicalized statistical parsing models,
- we develop a method to employ Expectation-Maximization on a smoothed parsing model and, using that method, show that heuristics used in several top-performing parsing models can be improved upon to form a better-estimated model,
- we build an extensible parsing engine and demonstrate accurate parsing models in English and Chinese, and develop models in other languages for the purpose of bootstrapping Treebank creation,
- we perform for the first time a detailed analysis of the parameters of a lexicalized statistical parsing model, and present which types of parameters are learned more and less well, and which types of parameters are more or less informative, which allows us to develop, for the first time, a robust parameter-selection method for a generative lexicalized statistical parsing model, and, finally,
- we perform the detailed parameter analysis on the Chinese model, enabling a novel type of comparison between the English and Chinese models and the treebanks on which they are trained.

The last two bullets are the most crucial: the “model as data” paradigm not only gives insight into the model itself that is under scrutiny, but also allows us to use the model as a lens onto the data on which it is trained, putting certain features into sharp relief that would otherwise not be visible.

1.4 History-based models

All of the models explored in detail in this dissertation will be *history-based models*, as introduced for parsing by Black et al. (1992). Such models were originally called “history-based grammar models”, but their formulation is far more general than merely as a mechanism for grammar modeling. We present a formal definition here, with slightly more detail than is given in previous treatments (Black et al., 1992; Collins, 1999).

Let \mathcal{X} be a (possibly infinite) event space to be modeled. Assume that each $x \in \mathcal{X}$ may be decomposed into at least one sequence of smaller events or decisions $d_1, \dots, d_n \in \mathcal{D}$ such that $P(x) = P(\bigcap_{i=1}^n d_i)$, with $|\mathcal{D}|$ finite. By the chain rule of probabilities, it is also true that

$$P\left(\bigcap_{i=1}^n d_i\right) = \prod_{i=1}^n P(d_i|d_0, \dots, d_{i-1}), \quad (1.1)$$

where $d_0 = \Omega_{\mathcal{D}}$ (the universe). The sequence of previous decisions is called the *history*. In order to reduce the size of the space to be modeled, we introduce a history mapping function $\Phi : \mathcal{D}^* \rightarrow \mathcal{C}$ that maps the entire sequence of previous decisions into one of a finite number of history contexts, $c \in \mathcal{C}$. Thus, we assume that

$$\prod_{i=1}^n P(d_i|d_0, \dots, d_{i-1}) = \prod_{i=1}^n P(d_i|\Phi(d_0, \dots, d_{i-1})). \quad (1.2)$$

One of the uses of Φ is to introduce independence assumptions into the model. For example, if we have $\Phi : \mathcal{D}^* \rightarrow \mathcal{C}$ s.t. $\Phi(d_0, \dots, d_{i-1}) = d_{i-1}$, then $\mathcal{C} = \mathcal{D}$ and the model reduces to a Markov model. The function Φ can also map elements of the complete history into newly-defined supertypes. For example, if the subsets $D_1, \dots, D_m \subset \mathcal{D}$, $m < |\mathcal{D}|$ represent a partition of \mathcal{D} , then we could have $\Phi(d_0, \dots, d_{i-1}) = D_j$, $d_{i-1} \in D_j$, reducing the cardinality of the range of Φ from $|\mathcal{D}|$ to m as compared to the previous example. In the context of a Markov part-of-speech tagger, this supertype could represent, say, the set of all noun tags, $\{\text{NN, NNS, NNP, NNPS}\}$, to use the Penn Treebank notation, allowing a model to condition on the fact that the previously-generated tag was *some* kind of noun, without having to be more specific. Both of these uses of Φ are functionally equivalent, in that the sequence of previous decisions is being reduced to some more-general class.

Decomposition into decision sequences and history-mapping are natural operations for sequence generation/classification problems, such as n -gram language modeling or part-of-speech

tagging, where simple definitions of Φ as per the examples above often suffice for instantiating high-quality models. In the case of parsing, Φ is typically used to form a tree structure of dependencies that is related to the derivation of the tree being generated by the model. This point will be made more concrete in Chapter 3.

Conditional history-based models are a straightforward extension of the single- or joint-space model defined above. Assume we are trying to model $P(y|x)$ for x, y in some joint space $\mathcal{X} \times \mathcal{Y}$.

Let \mathcal{D}_X and \mathcal{D}_Y be disjoint decomposition spaces for the elements of the original spaces \mathcal{X} and \mathcal{Y} , respectively, such that for all $x, y \in \mathcal{X} \times \mathcal{Y}$, there exist disjoint decision sequences $d_{x,1}, \dots, d_{x,m} \in \mathcal{D}_X$ and $d_{y,1}, \dots, d_{y,n} \in \mathcal{D}_Y$ where

$$P(x) = P\left(\bigcap_{j=1}^m d_{x,j}\right)$$

and

$$P(y) = P\left(\bigcap_{i=1}^n d_{y,i}\right).$$

Let $\mathcal{D} = \mathcal{D}_X \cup \mathcal{D}_Y$. We then have $P(y|x) = P(\bigcap_{i=1}^n d_{y,i}|x)$, or, using the chain rule

$$P\left(\bigcap_{i=1}^n d_{y,i}|x\right) = \prod_{i=1}^n P(d_{y,i}|d_0, d_{x,1}, \dots, d_{x,m}, d_{y,0}, \dots, d_{y,i-1}), \quad (1.3)$$

where $d_0 = d_{y,0} = \Omega_{\mathcal{D}}$. The exact equivalence of (1.3) is approximated by using a history-mapping function $\Phi : \mathcal{D} \rightarrow \mathcal{C}$, as follows:

$$P\left(\bigcap_{i=1}^n d_{y,i}|x\right) = \prod_{i=1}^n P(d_{y,i}|\Phi(d_0, d_{x,1}, \dots, d_{x,m}, d_{y,0}, \dots, d_{y,i-1})). \quad (1.4)$$

In the simplest case, x is not decomposed at all, and $d_0 = x$, but the above formulation allows the mapping function to access elements of the (potentially complex) conditioning event. In the case of part-of-speech tagging or parsing, for example, the conditioning event may be the entire sequence of words to be parsed. Decomposition of this sequence allows conditioning on, say, individual words.

1.5 Summary of this document

In Chapter 2, we explore previous work in the area of statistical parsing, with particular emphasis on the history of the field as it relates to the current crop of high-performance models that have

dominated the recent history of statistical parsing research. Chapter 3 presents the reproduction of what we have taken to be our baseline model, which turned out not to be trivial. Chapter 4 then shows the design of our parsing engine, which is capable of instantiating a wide variety of parsing models, with back-pointers to Chapter 3 to illustrate how a particular model type can serve as a guide for designing a flexible-yet-efficient engine. Chapter 5 presents the extension of a core model to multiple languages, with particular emphasis on the Chinese model, as that is the non-English language for which the most data is available. Next, Chapter 6 shows our initial work at model analysis, investigating the sorts of dependencies that appear to be useful for our baseline model, and a method to improve upon the heuristics that are used to add information to the raw training trees. Chapter 7 builds on the analysis work of the previous chapter, providing a detailed view of the distributions of parsing models for both English and Chinese. Finally, Chapter 8 presents our conclusions and describes areas for future work.

Essentially, this document follows a trajectory of parsing model analysis at increasing levels of detail:

- Chapter 3: baseline model and identification of the core and periphery
- Chapters 4 and 5: building a software engine that separates the core and periphery, and exploiting it to parse in multiple languages
- Chapter 6: analysis of meta-parameter space
- Chapter 7: detailed parameter analysis

1.6 What this dissertation is not

This research is about analysis and methodology, not methods. Our aim is to use a new methodology for the investigation of parsing models, specifically to gain new insight into how they function. New models will be developed in the course of this investigation, and some will be better in one or more dimensions than our baseline, and some will be worse. The models that provide insight into functionality are considered evidence that the methodology is effective, but these new models are, from the point of view of this dissertation, simply evidence, and not the point of the thesis.

Chapter 2

Previous work

Stochastic grammars are grammars that assign probabilities to their string-structure pairs, and such grammars form the basis of much of the work in statistical parsing of the last decade. Some of the earliest analysis of the properties of such grammars, particularly of probabilistic CFGs (PCFGs), was performed by Booth and Thompson (1973). Booth and Thompson define a *proper grammar* as one where the probabilities for all productions for a particular nonterminal sum to one. Furthermore, they showed that a PCFG was *consistent* if and only if the sum of the probabilities of all possible word strings generated within i steps equals 1 as i goes to infinity, where a generation step was an expansion of every nonterminal in the current derivation step.

While the use of stochastic grammars in parsing research would prove fruitful, from the outset researchers were aware of the particular limitations of PCFGs, *viz.*, the context-freeness of rule productions and the lack of lexical sensitivity. For the most part, the review in this chapter describes the various ways in which work in statistical parsing and related areas overcame these limitations. It is, however, a skewed view of the history of statistical parsing, owing to the dominance of lexicalized PCFG-based models in the field.

2.1 Seminal Work

The era of modern statistical parsing did not truly begin until many years after Booth and Thompson’s work, with the work of Magerman and Marcus (1991) and Black, Lafferty, and Roukos (1992). Magerman and Marcus (1991) developed a probabilistic, k -best chart parser called *Pearl*. This model attempted to incorporate several information sources into a single probabilistic framework, occasionally using some rather ad-hoc techniques. For example, in order to reduce the penalty imposed by probabilistic grammars on theories containing a lot of structure, *Pearl* used an ad-hoc “mean probability of all derivations” approach (this is not wholly dissimilar from the subsequent work of Rens Bod). It also used one of the simplest possible smoothing schemes, where a fixed amount—in their case, 0.5—was added to all counts (this technique is often called *Laplace smoothing* or *padding*). A significant aspect of *Pearl* was that the underlying CFG productions were actually not estimated as though they were context-free, but rather given the context of their parent productions. When the scoring function computed the score for a particular production occurring in context, it also conditioned on part-of-speech trigrams, but in a rather unobtrusive way.¹ While it is not possible to compare *Pearl*’s parsing performance to modern statistical parsers that operate over data for which we have gold-standard, hand-annotated parse trees, it provided an important stepping stone toward future research.

The four, coordinated areas of the research program of Black, Lafferty, and Roukos (1992), were seminal in several respects. First, the authors decided to work in a specific domain (computer manuals), that was reasonably broad in terms of sentence types and yet not completely unrestricted. This choice would be similar to that made by Marcus et al. (1993), who eschewed selecting a balanced set of data like the Brown corpus (Francis and Kučera, 1979) in favor of targeting Wall Street Journal sentences, the complexity and variety of which would still keep the problem interesting and meaningful. Second, Black et al. decided to develop a database of hand-parsed sentences—a treebank—where the annotators only marked up structure that could be determined in a consistent and reliable manner. These “skeletal” parses thus provided a degree of consistency for partially-supervised statistical learning methods that had theretofore had been difficult to attain. Third, they

¹The authors even include a footnote indicating that the part-of-speech trigram scoring function is more complicated than the way it is described in their paper.

separated the linguistic task of grammar writing from the statistical task of tree-structure modeling. While grammar-writing *per se* is no longer a pillar of parsing research, in their grammar Black et al. crucially employed syntactic labels that were actually feature bundles including, *inter alia*, “semantic” categories of phrasal heads. This idea, which most likely came from Ezra Black himself,² essentially put head-modifier relations (via syntactic categories) front-and-center, for the other researchers, being experts in pattern recognition, would model whatever data the linguists would present to them. Either directly or indirectly, this idea would spawn an entire line of research devoted to head-modifier relations. Finally, in the area of statistical modeling, Black et al. used a variant of the Inside-Outside algorithm (Lari and Young, 1990) that involved “constrained training” on their partially-bracketed corpus. This idea, which was also used and described formally in the contemporaneous research of Pereira and Schabes (1992), inspired several research efforts, including that of Hwa (1998), which in turn inspired the EM experiments of Chiang and Bikel (2002), described in Chapter 6.

Black et al. (1992) continued their seminal work by developing a *history-based grammar* (HBG) parsing model. This ground-breaking model could in theory handle any probabilistic grammar that had a well-ordered derivation, as it allowed a model of that grammar to predict the next output element conditioning on any previous aspect of the derivation. In this way, for example, one could take a PCFG and create a model with lexical dependencies, by conditioning future rule expansions on previously-generated lexical items. Indeed, Black et al. did this by explicitly lexicalizing the underlying PCFG grammar, associating no fewer than two heads with every nonterminal, a lexical head and a functional head. When estimating the likelihood of a given lexicalized nonterminal, their HBG model also employed two types of parents, an immediate parent and a functional parent, the latter for the case when the current child is the product of a unary production. In the computer manuals domain in which the IBM researchers were operating, their HBG model performed with significantly higher accuracy than a simple PCFG model. Moreover, this model was remarkably similar to the models of Collins (1996; 1997; 1999) that have seen such widespread use today.

In a separate effort from Black et al., a treebank was being developed in the early 1990’s at the

²Ezra Black was the prominent linguist among the group of IBM researchers with whom he worked.

University of Pennsylvania (Marcus, Santorini, and Marcinkiewicz, 1993).³ In spite of some now-well-known idiosyncrasies, this treebank was made widely available (although it has never been free), and, particularly in its second incarnation (“Treebank II”), the quality and consistency of the annotation was high enough to provide the springboard for virtually all of the recent successes in statistical parsing. By becoming a *de facto* standard, the Penn Treebank enabled parser researchers to compare the results of their systems (although this may also be causing some problems, due to implicit, human-driven overfitting via repeated testing on the same data).

2.2 Lexical relations

In a somewhat separate yet related line of research, Sekine et al. (1992), using head-modifier relations, developed a way to automatically form word clusters based on co-occurrence statistics. The idea was that words belonging to similar ontological classes tend to occur in similar linguistic contexts, so that by merely measuring the frequency with which words occur in the same (or similar) head-category-modifier relations, an algorithm can automatically compute a similarity between any two modifier or head words. In other words, co-occurrence statistics serve as a proxy for ontological similarity or semantic “distance”. The authors rightly observed that co-occurrence statistics could contain accidental frequencies where, *e.g.*, two semantically-unrelated words would happen to co-occur in the same linguistic context. The assumption was that “intrinsic” relations would be more frequent than such “accidental” ones.

In a similar vein, Alshawi and Carter (1994) investigated the use of several methods of determining lexical preference functions. They further investigated an application of these methods, comparing the utility of preference functions of lexical semantic collocations (essentially, various types of head-modifier relations). The authors hand-coded certain lexical classes, and used mutual information, chi-squared, likelihood ratio and other techniques to estimate semantic preferences of lexical items (or lexical classes) in the context of head-modifier relations, in order to score quasi-logical forms (QLFs) of sentences in the ATIS domain.

³While the IBM and Penn treebanking efforts were technically separate, it was the IBM researchers who conveyed the necessity of building a treebank to DARPA, who funded the creation of the Penn Treebank (Mitch Marcus, p.c.).

Parsing has for some time been concerned with the issue of head-modifier relations. Indeed, until recently, it was thought that almost all statistical parsers made heavy use of such “bilexical” relations, but this has turned out not to be the case, as will be discussed in detail in Chapters 6 and 7.

2.3 Stochastic lexicalized tree-adjoining grammars

In yet another line of research aimed at overcoming the limitations of PCFGs, particularly the lack of lexical sensitivity, Resnik (1992) and Schabes (1992) contemporaneously found ways to make a lexicalized tree-adjoining grammar (Schabes, 1990) into a stochastic grammar, and for estimating the associated probabilities. Rather than directly associate probabilities with derivation steps in an LTAG, Schabes uses the fact that for every LTAG, there is a lexicalized linear indexed grammar (LIG) that generates the same language with a one-to-one mapping between LTAG and lexicalized LIG derivations, as shown by Vijay-Shanker and Weir (1993). Schabes goes on to show a variant of CKY for parsing using a stochastic LIG (SLIG), as well as a variant of the Inside-Outside reestimation algorithm for SLIG. While a direct comparison with current work in parsing is not possible, Schabes illustrates how the Inside-Outside algorithm on his version of SLIG converges more quickly when reproducing the constrained training experiments on the ATIS corpus from Pereira and Schabes (1992). In contrast to Schabes, Resnik (1992) assigns probabilities directly to derivation operations—substitutions and adjunctions—as well as to the set of initial trees. Thus, the probability of a particular derivation is the product of probabilities of the initial tree and all derivation operations.

The above paved the way for much future work using stochastic TAGs and variants thereof. Xia (1999) devised a method for extracting a lexicalized TAG from the Penn Treebank and other, bracketed corpora. Sarkar (Sarkar, 2001; Sarkar, 2002) investigated ways to combine labeled and unlabeled data using stochastic LTAGs and a variant of Co-Training. Hwa (1998) explored a variant of LTAGs called lexicalized tree insertion grammars, which paved the way for Chiang’s work using the same formalism (Chiang, 2000; Chiang and Bikel, 2002).

2.4 Dependency parsing

In a line of research intended to place maximum emphasis on lexical items, Lafferty, Sleator, and Temperley (1992) exploit (what was then) a new formalism known as *link grammar*, where links are basically labeled arcs between two words in a sentence, where the label indicates the relation between the words, and where no two links are allowed to cross in a proper linkage (or parse) of a sentence. The authors develop a probabilistic treatment of this formalism, and then illustrate a finite-state approximation, showing how n -gram language models can be rather seamlessly integrated into a probabilistic link grammar. Finally, the authors address the problem of parameter estimation by developing yet another variant of the Inside-Outside algorithm.

Eisner (1996) notes that the parameterizations given by Lafferty et al. seem to capture “incidental properties of the data” rather than linguistically meaningful structures, and as such, would be likely to suffer from high entropy and/or over-fitting. He develops three lexico-centric probabilistic parsing models, and performs some preliminary evaluation using a small test set from the Penn Treebank. Model A models a random process that first assigns parts of speech to words (generates words and tags), and then tries to link pairs of words. Model B generates words and their parts of speech, and then simply generates each word-tag pair’s disjunct (set of selectional preferences); in the case where this scheme leads to a non-parse, the sentence is thrown out and probabilities are re-normalized. In Eisner’s Model C, the process is a head-driven recursive one, where first a head is generated along with its left- and right-side requirements, and then those requirements (word-tag pairs) are generated and the process recurses. Perhaps not surprisingly (in hindsight), this model yielded the highest accuracy in Eisner’s preliminary experiments.

2.5 The current era

The parsing problem was “blown wide open”⁴ by Magerman (1994). Magerman gained a great deal of experience in the area of pattern recognition at IBM, and used this experience to produce a conditional model of $p(\mathbf{T}|\mathbf{S})$, where \mathbf{T} is a tree derivation and \mathbf{S} is a sentence (word sequence). The

⁴Mitch Marcus (p.c.).

premise of Magerman's thesis was to have as little knowledge engineering as possible creep into the parsing model itself, instead relying on the data itself—the treebank—to contain all linguistic knowledge. Nevertheless, it is still necessary to choose features of the data to model, and Magerman chose features that were immediately obvious to him, and encoded and estimated most of those using statistical binary decision trees. A stack decoder (Jelinek, 1997, pp. 99ff.) was used to decode the overall model. With the vast array of statistical machinery and with a fairly large model, the process of training and decoding was complicated and computationally expensive, but the results were a quantum leap above existing parsing technologies.

In a much simpler approach, Collins (1996) defined a conditional model that used relative-frequency estimates of all probabilities. The cited aspect of this model was its reliance on lexical bigrams—the type of head-modifier relations that had previously been shown to be effective for NL disambiguation problems, including parsing. It was, perhaps, the particular parameterization that proved to be effective for parsing the Penn Treebank. Collins went on to define a cleaner, generative (joint) model (1997), that had even better performance on the now-standard Penn Treebank. It is this model, along with the subtle modifications described in Collins' thesis (1999) that we have taken to be our baseline model, as it remains one of the top-performing individual parsing models.

Contemporaneously with Collins, Ratnaparkhi (1997) developed a stochastic shift-reduce parser, where each reduce step was modeled by a maximum-entropy (log-linear) model. Not only did this model have accuracy comparable to Collins' 1997 model, but it also appeared to parse in time linear to the length of the input sentence. An additional advantage of the log-linear modeling technique was that it permitted the use of non-independent features.

While the initial properties of standard, unlexicalized PCFGs date back to Booth and Thompson (1973), results on the standard English Treebank test set were not available until Charniak (Charniak, 1996; Charniak, 1997). One very troubling aspect of these two papers is that they listed very different accuracy numbers for using PCFGs on the English Penn Treebank. In (Charniak, 1996), with a 30k-word test set (it is not specified which section), the labeled recall for sentences of length 2–40 words is 80.4% and labeled precision 78.8%. By contrast, (Charniak, 1997), on the now-standard Section 23, labeled recall is reported to be 71.2% and labeled precision 75.3% for sentences of length ≤ 40 words. It is highly unlikely that differences in test data would account

for such a large performance gap. One hypothesis might be that the 1996 results are for *unlabeled* recall and precision, but the text of the paper would indicate otherwise. However, the main result of (Charniak, 1997) is for a probabilistic parser including lexical information. In that model, parsing proceeds in two steps: first, a set of candidate parses is generated using the standard CKY parsing algorithm for a CFG derived directly from the Penn Treebank. Next, a statistical model that includes lexical dependences (similar to the head-modifier relations of (Black et al., 1992; Collins, 1996)) computes the most likely parse from the set of candidates. The results are slightly better than those reported in (Collins, 1996), but ultimately not as high as those reported in (Collins, 1997).

In the continuing evolution of parsing technology, Goodman (1997b) developed a new formalism called *probabilistic feature grammars* that he claimed combined the best aspects of the techniques used by Black, Magerman and Collins. The accuracy of his parser, however, remained at or below the level of Magerman's.

In a rather different approach, Rens Bod pushed a technique he called “data-oriented parsing” or DOP (Bod, 1992; Bod, 1993). DOP is not a specific formalism, but rather a research imperative: a DOP model is one that tries to model how tree fragments of any size have combined based on observed data. In other words, given observed trees, one should break them apart in all possible ways, and then estimate probabilities associated with all possible ways of re-combining the tree fragments. The Catalan-number of possibilities this presents means that estimation becomes intractable. Bod (1992) limits the size of tree fragments considered, whereas in later work he employs Monte Carlo methods. The stated rationale behind the approach is to mimic human learning of grammar, since we are exposed to sentences and, Bod presumes, their structure. While this may be controversial, as a research agenda, DOP has yielded some interesting ways to model sentences and their structures, not to mention some very accurate parsers.

2.6 Parser analysis

Johnson (1998) explored several types of tree/grammar transformations and their effect on PCFG probability estimates derived from relative frequencies in the English Penn Treebank. In particular, Johnson looked at how the Penn Treebank represents NP- and VP-modification by PPs differently:

in the case of NPs, PPs are Chomsky-adjoined, whereas in the case of VPs, they are not. One transformation was to make all PP modifications be Chomsky adjoined, adding structure where necessary to PP-modified VPs. Another transformation was basically the reverse: to “flatten” all structures to resemble the ternary-branching PP-modified VPs of the Penn Treebank. A third transformation involved annotating all nodes with the parent labels, in order to provide a model with more context sensitivity while remaining in the CFG paradigm. Johnson explored both the theoretical aspect of how relative-frequency estimates of NP modification could vary according to the various transformations, as well as the empirical aspect involving how well a PCFG could predict Treebank structure when the input grammar underwent the transformation under investigation and the output parse was “de-transformed” for evaluation purposes. In particular, the parent annotation scheme yielded a marked increase in both precision and recall over a standard PCFG.

In a completely empirical study, Gildea (2001) implemented a version of Model 1 (the simplest of three models) from (Collins, 1997). While the re-implementation was necessarily approximate (see Chapter 3 for the reason why), Gildea found some interesting results. First, as to the stated purpose of the inquiry, Gildea found that accuracy varied widely according to test set. Specifically, he observed that his version of Collins’ model when trained the parser on the WSJ Penn Treebank data (as normal) but tested on data from the Brown corpus that had been annotated in Treebank II-style had significantly lower performance. This indicated the degree to which his model was sensitive to domain. Furthermore, training on both WSJ and Brown data yielded only marginal improvements in accuracy on held-out test sets from the WSJ and Brown corpora over training on matched data for the respective test sets. The other, startling result was that removing the bilexical statistics from his version of Collins’ Model 1 yielded a mere 0.5% drop in accuracy, indicating that bilexical statistics—once thought to be the main power of such models—were barely being used. (See Chapter 6 for more on this result, and on my additional results in this area.)

2.7 Recent work

By 1999, there were several statistical parsers that were operating in the 86–89% accuracy range on the Penn Treebank. Henderson and Brill (1999) decided to exploit the less-than-100% correlation

of these top-performing parsers' errors by combining them in various ways. Specifically, they used the parsers of Ratnaparkhi (1997), Charniak (1997) and Collins (1997), and devised three methods of what they called *parser hybridization*. The first method is called *constituent voting*, where each parser independently voted on each of the constituents in its maximally-likely parse of a sentence to be included in the hybrid parser's parse. An alternative, constituent-based scheme was to use a naïve-Bayes classifier to choose which constituents to include in the hybrid parser's parse. The third method was called *parser switching*, and was used to choose which of the candidate parses to use as a whole, which was useful in the cases where it was important to preserve entire analyses. The non-probabilistic version of this technique simply chose the parse that was closest to the centroid of all the candidate parses, and the probabilistic version used a naïve-Bayes classifier. The experimental results for these techniques were impressive: using either of the constituent-based combining techniques, labeled recall was 90.10% and labeled precision was 92.42% on the held-out test set, which was Section 22 of the English Penn Treebank. Given an oracle that could perform optimal parser switching, labeled recall was 93.87% and labeled precision was 93.78%. These results showed that individual models had a long way to go before they reached an upper bound, or, conversely, that extremely high-accuracy parsing was possible, even with the vagaries of the Penn Treebank.

On the front of particular parsing models, Charniak updated his earlier model with one that included many additional features (Charniak, 2000). He called this a “maximum entropy–inspired parser”, and yet the actual parser had nothing to do with maximum entropy or log-linear models. In fact, in his oral presentation at the NAACL2000 conference, he noted that the single greatest performance improvement was due to predicting nonterminals and their head word's part-of-speech tag in one step, following Collins (1997). In spite of the mathematical oddities of the model, it achieved impressive accuracy results of 90.1% labeled recall/precision on sentences of length ≤ 40 words on Section 23 of the Penn Treebank.

Collins, noticing that his generative model could not easily capture long-range dependencies, devised a method to perform re-ranking of subtrees proposed by a k -best version of his generative model, using long-range features (Collins, 2000). The basic idea was to learn parameter settings for his long-range parameters that minimized various loss functions. For score-obsessed parsing

fanatics, the results were every bit as impressive as Charniak's from the previous year: 90.1% labeled recall and 90.4% labeled precision on Section 23. The bulk of (Collins, 2000) focuses on the loss functions and optimization methods, and the end of the paper lists the long-range features employed. Many of these were meant to deal with the always-thorny problems of PP attachment and coordination, using features that had proved successful in earlier efforts to solve those smaller, individual classification problems, but were difficult to incorporate into a history-based, generative model.

Following on the work of Resnik (1992) and Schabes (1992), Chiang (2000) developed a stochastic lexicalized tree-insertion grammar (SLTIG) parser. This model bore a great deal of intentional resemblance to that of Collins (1999). The idea of this work was twofold: by producing an accurate LTIG parser, one would have access not only to the highest-probability derived tree, which would be needed for comparative evaluation purposes, but also to the *derivation tree*, which would provide a richer type of dependency information. Also, it was hoped that the larger domain of locality provided by LTIG would yield more accurate parsing results than models with an underlying lexicalized PCFG. Unfortunately, parsing the Penn Treebank is a tricky business, and so Chiang's accuracy results, while competitive with all the recent work, were not significantly higher as originally hoped. However, Chiang's LTIG parser proved to be a useful framework for additional experimentation (see (Chiang and Bikel, 2002) and Chapter 6 of this document).

Most recently, Klein and Manning (2003) developed a largely unlexicalized PCFG that outperforms the lexicalized model of Magerman (1995). The approach used two main insights: first, drawing from the parent-annotation results of Johnson (1998), Klein et al. used a Markov model on the *vertical* path set of a nonterminal, in addition to a now-familiar Markovization of the children of a particular production. Second, Klein et al. observed that Penn Treebank part-of-speech tags were often not fine-grained enough to allow an unlexicalized parsing model to disambiguate competing syntactic analyses. Thus, while not truly lexicalizing the underlying grammar or parsing model, the researchers chose to augment certain tags with words that they claimed were closed-class functional heads.⁵ The resulting parser achieved an impressive labeled recall of 85.7% and a labeled precision

⁵One might take exception to calling forms of the verb *to be* as functional heads, and yet some of these were included in the tag augmentations.

of 86.9% on sentences of length ≤ 40 words.

2.8 Relation to the present work

By adopting the model of Collins (1999) as a baseline, the present work sits squarely in the successful tradition begun largely by Black, Lafferty, and Roukos (1992), employing a history-based model with head-modifier dependencies. However, a baseline model is usually one that is either simple, obvious or both, and Collins' thesis model proved to be neither, as described in detail in Chapter 3. Chapter 4 describes the design of our modular, extensible parsing engine, and shows how that design was influenced in part by the intricacies detailed in Chapter 3, crucially separating the core of the model from the part that is language-dependent. Chapter 5 explores extending the core model to other languages, following a line of research begun in (Bikel and Chiang, 2000), and related to the more recent work of Hwa, Resnik, and Weinberg (2002). Chapter 6 discusses and analyzes alternative dependencies that follow from the work of Johnson (1998), Collins (2000) and Charniak (2000); this chapter also presents a new variant of the Inside-Outside EM method for parsing models, building on previous work in that area (Lari and Young, 1990; Pereira and Schabes, 1992; Hwa, 1998). Finally, Chapter 7 builds on this type of analysis, but at a much greater level of detail than has been previously explored. In both Chapters 6 and 7, the use of bilexical dependencies is explored in great detail, which is related to the previous work in the area of stochastic encoding of lexical relations (Sekine et al., 1992; Lafferty, Sleator, and Temperley, 1992; Eisner, 1996).

Chapter 3

Reproducing the baseline model: a complicated task

Michael Collins' parsing models (Collins, 1996; Collins, 1997; Collins, 1999) have been quite influential in the field of natural language processing. Not only did they achieve new performance benchmarks on parsing the Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993), and not only did they serve as the basis of Collins' own future work (Collins, 2000; Collins and Duffy, 2002), but they also served as the basis of important work on parser selection (Henderson and Brill, 1999), an investigation of corpus variation and the effectiveness of bilexical dependencies (Gildea, 2001), sample selection (Hwa, 2001), bootstrapping non-English parsers (Hwa, Resnik, and Weinberg, 2002) and for the automatic labeling of semantic roles and predicate-argument extraction (Gildea and Jurafsky, 2000; Gildea and Palmer, 2002), as well as that of other research efforts.

Recently, in order to continue our work combining word sense with parsing (Bikel, 2000) and the study of language-dependent and -independent parsing features (Bikel and Chiang, 2000), we built a *multi-lingual parsing engine* that is capable of instantiating a wide variety of generative, statistical parsing models (Bikel, 2002).¹ As an appropriate baseline model, we chose to instantiate the parameters of Collins' Model 2. This task proved more difficult than it initially appeared. Starting with Collins' thesis (Collins, 1999), we reproduced all the parameters described, but did not

¹This engine is publicly available by visiting <http://www.cis.upenn.edu/~dbikel/software.html>.

achieve nearly the same high performance on the well-established development test set of Section 00 of the Penn Treebank. Together with Collins' thesis, this chapter contains all the information necessary to replicate Collins' parsing results.² Specifically, this chapter describes all the as-yet-unpublished details and features of Collins' model, and some analysis of the effect of these features with respect to parsing performance, as well as some comparative analysis of the effects of published features.³ In particular, implementing Collins' model using only the published details causes an 11% increase in relative error over Collins' own published results. That is, taken together, all the unpublished details have a significant effect on overall parsing performance. In addition to the effects of the unpublished details, we also have new evidence to show that the discriminative power of Collins' model does not lie where once thought: bilexical dependencies play an extremely small role in Collins' models (Gildea, 2001), and head choice is not nearly as critical as once thought. This chapter also discusses the rationale for various parameter choices. In general, we will limit our discussion to Model 2, but we make occasional reference to Model 3, as well.

3.1 Motivation

There are three primary motivations for this work. First, Collins' parsing model represents a widely-used and -cited parsing model. As such, if it is not desirable to use it as a black box (it has only recently been made publicly available), then it should be possible to replicate the model in full, providing a necessary consistency among research efforts employing it. Careful examination of its intricacies will also allow researchers to deviate from the original model when they think it is warranted, and accurately document those deviations, as well as understand the implications of doing so.

The second point is related to the first: science dictates that experiments be replicable, for this

²In the course of replicating Collins' results, it was brought to our attention that several other researchers had also tried to do this, and had also gotten performance that fell short of Collins' published results. For example, Gildea (2001) re-implemented Collins' Model 1, but obtained results with roughly 16.7% more relative error than Collins' reported results using that Model.

³Discovering these details and features involved a great deal of reverse-engineering, and, ultimately, much discussion with Mike Collins himself and perusal of his code. Many thanks to Mike Collins for his generosity. As a word of caution, this chapter is exhaustive in its presentation of all such details and features, and we cannot guarantee that every reader will find every detail interesting.

is the way we may test and validate them. The work described here comes in the wake of several previous efforts to replicate this particular model, but this is the first such effort to provide a faithful and equally-well-performing emulation of the original.

The third motivation is that a deep understanding of an existing model—its intricacies, the interplay of its many features—provides the necessary platform for advancement to newer, “better” models. This is especially true in an area like statistical parsing that has seen rapid maturation followed by a “soft ceiling” in performance. Rather than simply throwing features into a new model and measuring their effect in a crude way by using standard evaluation metrics, this work aims to provide a more thorough understanding of the *nature* of a model’s features. This understanding is not only useful in its own right, but should help point the way toward newer features to model, or better modeling techniques, for we are in the best position for advancement when we understand existing strengths and limitations.

3.2 Model Overview

The Collins parsing model decomposes the generation of a parse tree into many, small steps, using reasonable independence assumptions to make the parameter estimation problem tractable. Even though decoding proceeds bottom-up, the model is defined in a top-down manner. Every nonterminal label in every tree is *lexicalized*: the label is augmented to include a unique head word (and that head word’s part of speech) that the node dominates. The lexicalized PCFG that sits behind Model 2 has rules of the form

$$P \rightarrow L_n L_{n-1} \cdots L_1 H R_1 \cdots R_{n-1} R_n \quad (3.1)$$

where P , L_i , R_i and H are all lexicalized nonterminals, and P inherits its lexical head from its distinguished head child, H . In this generative model, first P is generated, then its head-child H , then each of the left- and right-modifying nonterminals are generated from the head outward. The modifying nonterminals L_i and R_i are generated conditioning on P and H , as well as a distance metric (based on what material intervenes between the currently-generated modifying nonterminal and H) and an incremental subcategorization frame feature (a multiset containing the arguments of H that have yet to be generated on the side of H in which the currently-generated nonterminal

falls). Note that if the modifying nonterminals were generated completely independently, the model would be very impoverished, but in actuality, by including the distance and subcategorization frame features, the model captures a crucial bit of linguistic reality, *viz.*, that words often have well-defined sets of complements and adjuncts, occurring with some well-defined distribution in the right hand sides of a (context-free) rewriting system. The process proceeds recursively, treating each newly-generated modifier as a parent and then generating its head and modifier children; the process terminates when (lexicalized) preterminals are generated. As a way to guarantee the consistency of the model, the model also generates two hidden +STOP+ nonterminals as the leftmost and rightmost children of every parent (see Figure 3.7 in §3.4.1).

3.3 Preprocessing training trees

To the casual reader of Collins' thesis, it may not be immediately apparent that there are quite a few preprocessing steps for each annotated training tree, and that these steps are crucial to the performance of the parser. We identified eleven preprocessing steps necessary to prepare training trees when using Collins' parsing model. They are:

1. pruning of unnecessary nodes
2. adding base NP nodes (NPBs)
3. "repairing" base NPs
4. adding gap information (applicable to Model 3 only)
5. relabeling of sentences with no subjects (subjectless sentences)
6. removing null elements
7. raising punctuation
8. identification of argument nonterminals
9. stripping unused nonterminal augmentations

10. “repairing” subjectless sentences

11. head-finding

The order presented above is not arbitrary, as some of the steps depend on results produced in previous steps. Also, we have separated the steps into their functional units; an implementation could combine steps that are independent of one another (for clarity, our implementation does not, however). Finally, we note that the final step, head-finding, is actually needed by some of the previous steps in certain cases; in our implementation, we selectively employ a head-finding module during the first 10 steps where necessary.

3.3.1 Coordinated phrases

A few of the preprocessing steps rely on a notion of a coordinated phrase. The conditions under which a phrase is considered coordinated are slightly more detailed than described in Collins’ thesis. A node represents a coordinated phrase if

- it has a non-head child that is a coordinating conjunction, and
- that conjunction is either
 - post-head but non-final, or
 - immediately pre-head but non-initial (where “immediately” means “with nothing intervening except punctuation”).⁴

In the Penn Treebank, a coordinating conjunction is any preterminal node with the label CC.

This definition essentially picks out all phrases where the head child is truly conjoined to some other phrase, as opposed to a phrase where, say, there is an initial CC word, such as an S that begins with the conjunction “But . . .”.

⁴Our positional descriptions here, such as “post-head but non-final”, refer to positions within the list of immediately-dominated children of the coordinated phrase node, as opposed to positions within the entire sentence.

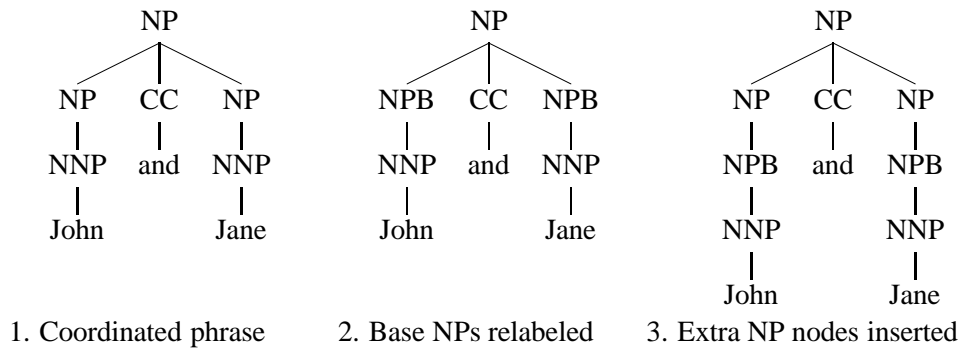


Figure 3.1: An NP that constitutes a coordinated phrase.

3.3.2 Pruning of unnecessary nodes

This preprocessing step simply removes preterminals that should have little or no bearing on parser performance. In the case of the English Treebank, the pruned subtrees are all preterminal subtrees whose root label is one of {", " , .}. There are two reasons to remove these types of subtrees when parsing the English Treebank: in the treebanking guidelines (Bies, 1995), quotation marks were given the lowest possible priority, and thus cannot be expected to appear within constituent boundaries in any kind of consistent way, and neither of these types of preterminals—nor *any* punctuation marks, for that matter—count towards the parsing score.

3.3.3 Adding base NP nodes (NPBs)

An NP is *basal* when it does not itself dominate an NP; such NP nodes are relabeled NPB. More accurately, an NP is basal when it dominates no other NPs except possessive NPs, where a possessive NP is an NP that dominates POS, the preterminal possessive marker for the Penn Treebank. These possessive NPs are almost always themselves base NPs, and are therefore (almost always) relabeled NPB.

For consistency’s sake, when an NP has been relabeled as NPB, a normal NP node is often inserted as a parent nonterminal. This insertion ensures that NPB nodes are always dominated by NP nodes. The conditions for inserting this “extra” NP level are slightly more detailed than described in Collins’ thesis, however. The extra NP level is added if one of the following conditions holds:

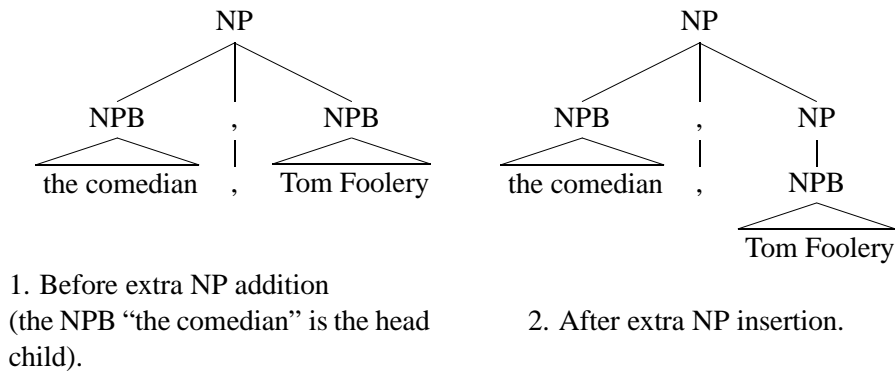


Figure 3.2: A non-head NPB child of NP requires insertion of extra NP.

- the parent of the NPB is not an NP
- the parent of the NPB *is* an NP, but constitutes a coordinated phrase (see Figure 3.1)
- the parent of the NPB *is* an NP, but
 - the parent’s head child is not the NPB and
 - the parent has not already been relabeled as an NPB (see Figure 3.2)⁵

In post-processing, when an NPB is an only child of an NP node, the extra NP level is removed by merging the two nodes into a single NP node, and all remaining NPB nodes are relabeled NP.

3.3.4 Repairing base NPs

The insertion of extra NP levels above certain NPB nodes achieves a degree of consistency for NPs, effectively causing the portion of the model that generates children of NP nodes to have less perplexity. Collins appears to have made a similar effort to improve the consistency of the NPB model. NPB nodes that have sentential nodes as their final (rightmost) child are “repaired”: the sentential child is raised so that it becomes a new right-sibling of the NPB node (see Figure 3.3).⁶

⁵Only applicable if relabeling of NPs is performed using a pre-order tree traversal.

⁶Collins defines a sentential node, for the purposes of repairing NPBs, to be any node that begins with the letter S. For the Penn Treebank, this defines the set {S, SBAR, SBARQ, SINV, SQ}.

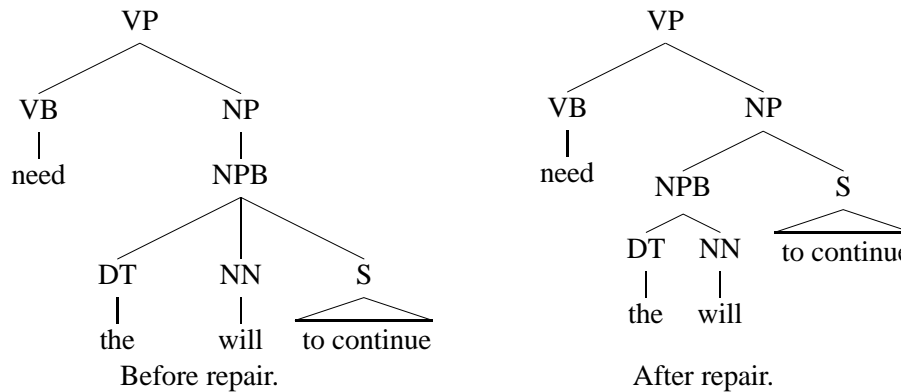


Figure 3.3: An NPB is “repaired”.

While such a transformation is reasonable, it is interesting to note that Collins’ parser performs no equivalent de-transformation when parsing is complete, meaning that when the parser produces the “repaired” structure during testing, there is a spurious NP bracket.⁷

3.3.5 Adding gap information

The gap feature is discussed extensively in chapter 7 of Collins’ thesis, and is applicable only to his Model 3. This preprocessing step locates every null element preterminal, finds its co-indexed WHNP antecedent higher up in the tree, replaces the null element preterminal with a special trace tag and threads the gap feature in every nonterminal in the chain between the common ancestor of the antecedent and the trace. The only detail we would like to highlight here is that an implementation of this preprocessing step should check for cases where threading is impossible, such as when two filler-gap dependencies cross. An implementation should be able to handle nested filler-gap dependencies, however.

3.3.6 Relabeling subjectless sentences

The node labels of sentences with no subjects are transformed from S to SG. This step enables the parsing model to be sensitive to the different contexts in which such subjectless sentences occur as compared to normal S nodes, since the subjectless sentences are functionally acting as noun

⁷Since, as mentioned above, the only time an NPB is merged with its parent is when it is the only child of an NP.

phrases. Collins' example of

[_S [_S Flying planes] is dangerous]

illustrates the utility of this transformation. However, the conditions under which an S may be relabeled are not spelled out; one might assume that every S whose subject (identified in the Penn Treebank with the -SBJ function tag) dominates a null element should be relabeled SG. In actuality, the conditions are much stricter. An S is relabeled to SG when the following conditions hold:

- one of its children dominates a null element child marked with -SBJ
- its head child is a VP
- no arguments appear prior to the head child (see §§3.3.9 and 3.3.11)

The latter two conditions appear to be an effort to capture only those subjectless sentences that are based around gerunds, as in the “flying planes” example.⁸

3.3.7 Removing null elements

This step simply involves pruning the tree to eliminate any subtree that only dominates null elements. The special trace tag that is inserted in the step that adds gap information (§3.3.5) is excluded, as it is specifically chosen to be something other than the null element preterminal marker (which is -NONE- in the Penn Treebank).

3.3.8 Raising punctuation

This step is discussed in detail in chapter 7 of Collins' thesis. The main idea is to raise punctuation—which is any preterminal subtree where the part of speech is either a comma or a colon—to the highest possible point in the tree, so that it always sits between two other nonterminals. Punctuation that occurs at the very beginning or end of a sentence is “raised away”, *i.e.*, pruned. In

⁸We assume the “G” in the label SG was chosen to stand for the word “gerund”.

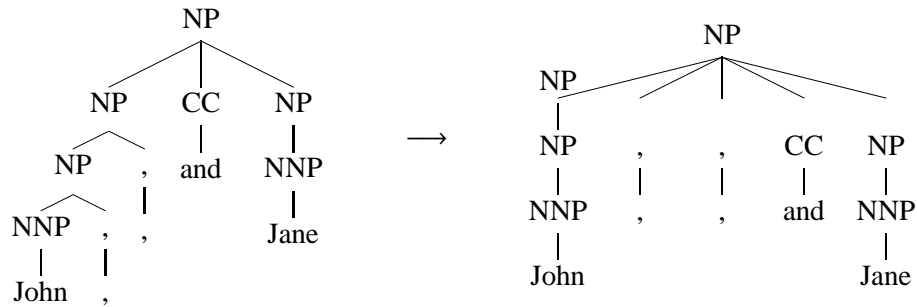


Figure 3.4: Raising punctuation: perverse case where multiple punctuation elements appear along a frontier of a subtree.

addition, any implementation of this step should handle the case where multiple punctuation elements appear as the initial or final children of some node, as well as the more pathological case where multiple punctuation elements appear along the left or right frontier of a subtree (see Figure 3.4). Finally, it is not clear what to do with nodes that *only* dominate punctuation preterminals. Our implementation simply issues a warning in such cases, and leaves the punctuation symbols untouched.

3.3.9 Identification of argument nonterminals

Collins employs a small set of heuristics to mark certain nonterminals as arguments, by appending -A to the nonterminal label. This section reveals three unpublished details about Collins' argument finding.

- The published argument-finding rule for PPs is to choose the first nonterminal after the head child. In a large majority of cases, this marks the NP argument of the preposition. The actual rule used is slightly more complicated: the first nonterminal to the right of the head child that is neither PRN nor a part of speech tag is marked as an argument. The nonterminal PRN in the Penn Treebank marks parenthetical expressions, which can occur fairly often inside a PP, as in the phrase “on (or above) the desk”.
- Children that are part of a coordinated phrase (see §3.3.1) are exempt from being relabeled as argument nonterminals.

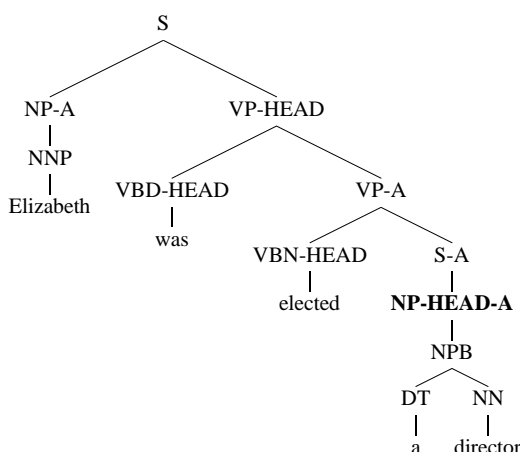


Figure 3.5: Head children are not exempt from being relabeled as arguments.

- Head children are distinct from their siblings by virtue of the head-generation parameter class in the parsing model. In spite of this, Collins’ trainer actually does *not* exempt head children from being relabeled as arguments (see Figure 3.5).⁹

3.3.10 Stripping unused nonterminal augmentations

This step simply involves stripping away all nonterminal augmentations, except those that have been added from other preprocessing steps (such as the *-A* augmentation for argument labels). This includes the stripping away of all function tags and indices marked by the Treebank annotators.

3.3.11 Repairing subjectless sentences

With arguments identified as described in §3.3.9, if a subjectless sentence is found to have an argument prior to its head, this step de-transforms the SG so it reverts to being an S.

3.3.12 Head-finding

Head-finding is discussed at length in Collins’ thesis, and the head-finding rules used are included in his Appendix A. There are a few unpublished details worth mentioning, however.

⁹It is not clear why this is done, and so in our parsing engine, we make such behavior optional via a run-time setting.

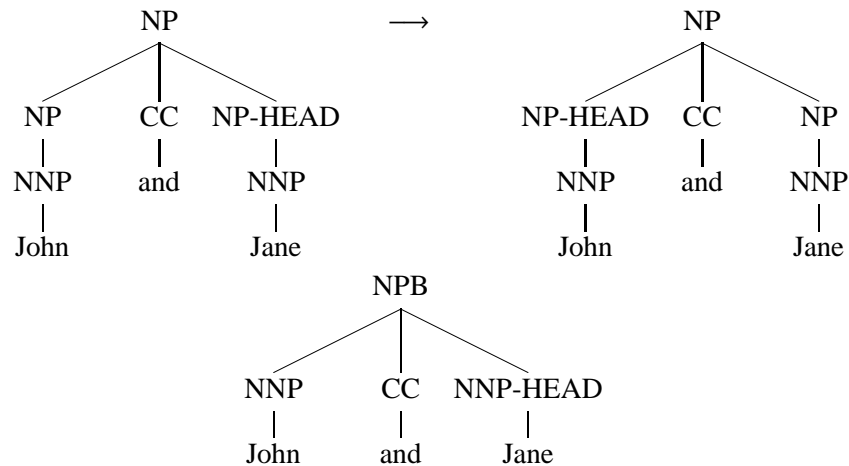


Figure 3.6: Head moves from right to left conjunct in a coordinated phrase, *except* when the parent nonterminal is NPB.

There is no head-finding rule for NX nonterminals, so the default rule of picking the leftmost child is used.¹⁰ NX nodes roughly represent the N' level of syntax, and in practice often denote base NPs. As such, the default rule often picks out a less-than-ideal head child, such as an adjective that is the leftmost child in a base NP.

Collins' thesis discusses a case where the initial head is modified when it is found to denote the right conjunct in a coordinated phrase. That is, if the head rules pick out a head that is preceded by a CC that is non-initial, the head should be modified to be the nonterminal immediately to the left of the CC (see Figure 3.6). An important detail is that such “head movement” does *not* occur inside base NPs. That is, a phrase headed by NPB may indeed *look* like it constitutes a coordinated phrase—it has a CC that is non-initial but to the left of the currently-chosen head—but the currently-chosen head should remain as is.¹¹ As we shall see, there is exceptional behavior for base NPs in almost every part of the Collins parser.

¹⁰In our first attempt at replicating Collins' result, we simply employed the same head-finding rule for NX nodes as for NP nodes. This choice yields different—but not necessarily inferior—results.

¹¹In §3.3.1, we defined coordinated phrases in terms of heads, but here we are discussing how the head-finder itself needs to determine whether a phrase is coordinated. It does this by considering the potential new choice of head: if the head-finding rules pick out a head that is preceded by a non-initial CC (“Jane”), will moving the head to be a child to the left of the CC (“John”) yield a coordinated phrase? If so, then the head should be moved—except when the parent is NPB.

3.4 Training

The trainer’s job is decompose annotated training trees into a series of head- and modifier-generation steps, recording the counts of each of these steps. Referring to (3.1), each H , L_i and R_i are generated conditioning on previously-generated items, and each of these events consisting of a generated item and some maximal history context are counted. Even with all this decomposition, sparse data is still a problem, and so each probability estimate for some generated item given a maximal context is smoothed with coarser distributions using less context, whose counts are derived from these “top-level” head- and modifier-generation counts.

3.4.1 Verb intervening

As mentioned in §3.2, instead of generating each modifier independently, the model conditions the generation of modifiers on certain aspects of the history. One such function of the history is the *distance metric*. One of the two components of this distance metric is what we will call the “verb intervening” feature, which is a predicate vi that is `true` if a verb has been generated somewhere in the surface string of the previously-generated modifiers on the current side of the head. For example, in Figure 3.7, when generating the right-hand +STOP+ nonterminal child of the VP, the vi predicate is `true`, because one of the previously-generated modifiers on the right side of the head dominates a verb, “continue”.¹² The definition of this feature is most easily defined in terms of a recursively-defined cv (“contains verb”) predicate, which is `true` if and only if a node dominates a verb.

$$cv(P) = \begin{cases} \bigvee_{M \text{ child of } P} cv(M) & \text{if } P \text{ is not a preterminal,} \\ \text{true} & \text{if } P \text{ is a verb preterminal,} \\ \text{false} & \text{otherwise.} \end{cases} \quad (3.2)$$

¹²Note that *any* verb in the surface strings dominated by the previously-generated modifiers will trigger the vi predicate. This is possible because in a history-based model (*cf.* (Black et al., 1992)), anything previously-generated—*i.e.*, anything in the *history*—can appear in the conditioning context.

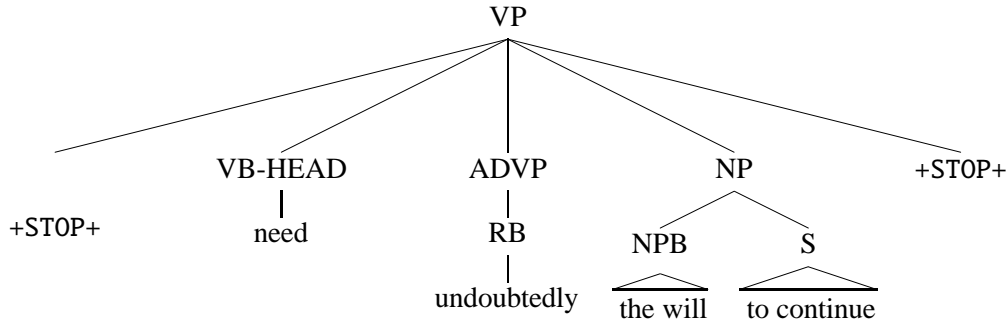


Figure 3.7: *vi* (“verb intervening”) feature is *true* when generating right-hand *+STOP+* nonterminal, because the NP “the will to continue” contains a verb.

Referring to (3.2), we define the verb intervening predicate recursively on the 1st-order Markov process generating modifying nonterminals:

$$vi(L_i) = \begin{cases} \text{false} & \text{if } i \leq 1, \\ cv(L_{i-1}) \vee vi(L_{i-2}) & \text{if } i > 1, \end{cases} \quad (3.3)$$

and similarly for right modifiers.

What is considered to be a verb? While this is not spelled out, as it happens, it is any word whose part-of-speech tag is one of $\{VB, VBD, VBG, VBN, VBP, VBZ\}$. That is, the *cv* predicate returns *true* only for these preterminals and *false* for all other preterminals. Crucially, this set omits *MD*, which is the marker for modal verbs. Another crucial point about the *vi* predicate is that it does *not* include verbs that appear within base NPs. Put another way, in order to emulate Collins’ model, we need to amend the definition of *cv* by stipulating that $cv(NPB) = \text{false}$.

3.4.2 Skip certain trees

One oddity of Collins’ trainer that we mention here for completeness’ sake is that it skips certain training trees. For “odd historical reasons”,¹³ the trainer skips all trees with more than 500 tokens, where in this context a token is considered to be a word, a nonterminal label or a parenthesis. This oddity entails that even some relatively short sentences get skipped because they have lots of tree structure. In the standard Wall Street Journal training corpus, Sections 02–21 of the Penn

¹³This phrase was taken from a comment in one of Collins’ preprocessing Perl scripts.

Trebank, there are 120 such sentences that are skipped. Unless there is something inherently wrong with these trees, one would predict that adding them to the training set would improve a parser’s performance. As it happens, there is actually a minuscule (and probably statistically insignificant) *drop* in performance (see §3.7, Table 3.5).

3.4.3 Unknown words

3.4.3.1 The threshold problem

Collins mentions in Chapter 7 of his thesis that “[a]ll words occurring less than 5 times in training data, and words in test data which have never been seen in training, are replaced with the ‘UNKNOWN’ token.” The frequency below which words are considered unknown is often called *the unknown word threshold*. Unfortunately, this term can also refer to the frequency *above* which words are considered known. As it happens, the unknown word threshold Collins uses in his parser for English is 6, not 5.¹⁴ To be absolutely unambiguous, words that occur *fewer than 6 times*, which is to say words that occur *5 times or fewer* in the data are considered “unknown”.

3.4.3.2 Not handled in a uniform way

The obvious way to incorporate unknown words into the parsing model, then, is simply to map all low-frequency words in the training data to some special +UNKNOWN+ token before counting top-level events for parameter estimation (where “low-frequency” means “below the unknown word threshold”). Collins’ trainer actually does not do this. Instead, it does not directly modify *any* of the words in the original training trees, and proceeds to break up these unmodified trees into the top-level events. After these events have been collected and counted, the trainer selectively maps low-frequency words when *deriving* counts for the various context (back-off) levels of the parameters that make use of bilinear statistics. If this mapping were performed uniformly, then it would be identical to mapping low-frequency words prior to top-level event counting; this is not the case, however. We describe the details of this unknown-word mapping in §3.5.8.2.

¹⁴As with many of the discovered discrepancies between the thesis and the implementation, we determined the different unknown word threshold by reverse-engineering, in this case by an analysis of the events output by Collins’ trainer.

While there is a negligible yet detrimental effect on overall parsing performance when one uses an unknown word threshold of 5 instead of 6, when this change is combined with the “obvious” method for handling unknown words, there is actually a minuscule improvement in overall parsing performance (see §3.7, Table 3.5).

3.5 Parameter Classes and Their Estimation

All parameters that generate trees in Collins’ model are estimates of conditional probabilities. Even though the following overview of parameter classes presents only the maximal contexts of the conditional probability estimates, it is important to bear in mind that the model always makes use of smoothed probability estimates that are the linear interpolation of several raw maximum-likelihood estimates, using various amounts of context (we will explore smoothing in detail in §3.5.7). Appendix B provides a complete listing of all parameter classes with their back-off contexts.

Mapped versions of the set of nonterminals

In §§3.3.5 and 3.3.9, we saw how the raw Treebank nonterminal set is expanded to include nonterminals augmented with $-A$ and $-g$. Although it is not made explicit in Collins’ thesis, Collins’ model uses two mapping functions to remove these augmentations when including nonterminals in the history contexts of conditional probabilities. Presumably this was done to help alleviate sparse data problems. We will notate the “argument removal” mapping function as α and the “gap removal” mapping function as γ . For example,

- $\alpha(\text{NP-A-g}) = \text{NP-g}$,
- $\gamma(\text{NP-A-g}) = \text{NP-A}$ and
- $\alpha(\gamma(\text{NP-A-g})) = \text{NP}$.

Since gap augmentations are only present in Model 3, the γ function effectively is the identity function in the context of Models 1 and 2.

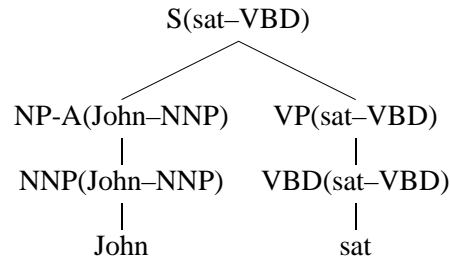


Figure 3.8: A fully lexicalized tree. The VP node is the head child of S.

3.5.1 The head parameter class

The head nonterminal is generated conditioning on its parent nonterminal label, as well as the head word and head tag which they share, since parents inherit their lexical head information from their head children. More specifically, an unlexicalized head nonterminal label is generated conditioning on the fully-lexicalized parent nonterminal. We notate the parameter class as follows:

$$P_H(H | \gamma(P), w_h, t_h) \quad (3.4)$$

3.5.2 The subcat parameter class

When the model generates a head child nonterminal for some lexicalized parent nonterminal, it also generates a kind of subcategorization frame (subcat) on either side of the head child, with the following maximal context:

$$P_{subcat_L}(subcat_L | \alpha(\gamma(H)), \alpha(\gamma(P)), w_h, t_h) \quad (3.5)$$

$$P_{subcat_R}(subcat_R | \alpha(\gamma(H)), \alpha(\gamma(P)), w_h, t_h) \quad (3.6)$$

Probabilistically, it is as though these subcats are generated with the head child, via application of the chain rule, but they are conditionally independent.¹⁵ These subcats may be thought of as lists of requirements on a particular side of a head. For example, in Figure 3.8, after the root node

¹⁵Using separate steps to generate subcats on either side of the head not only allows for conditional independence between the left and right subcats, but also allows for these parameters to be separately smoothed from the head-generation parameter.

of the tree has been generated (see §3.5.9), the head child VP is generated, conditioning on both the parent label S and the head word of that parent, *sat*-VBD. Before any modifiers of the head child are generated, both a left- and right-subcat frame are generated. In this case, the left subcat is {NP-A} and the right subcat is {}, meaning that there are no required elements to be generated on the right side of the head. Subcats do not specify the order of the required arguments. They are dynamically-updated multisets: when a requirement has been generated, it is removed from the multiset and subsequent modifiers are generated conditioning on the updated multiset.¹⁶

The implementation of subcats in Collins' parser is even more specific: subcats are multisets containing various numbers of precisely six types of items: NP-A, S-A, SBAR-A, VP-A, g and miscellaneous. The "g" indicates that a gap must be generated, and is applicable only to Model 3. Miscellaneous requirements include all nonterminals that were marked as arguments in the training data that were not any of the other named types. There are rules for determining whether NPs, Ss, SBARs and VPs are arguments, and the miscellaneous arguments occur as the result of the argument-finding rule for PPs, which states that the first non-PRN, non-part-of-speech tag that occurs after the head of a PP should be marked as an argument, and therefore nodes that are not one of the four named types can be marked.

3.5.3 The modifying nonterminal parameter class

As mentioned above, after a head child and its left and right subcats are generated, modifiers are generated from the head outward, as indicated by the modifier nonterminal indices in Figure 3.1. A fully-lexicalized nonterminal has three components: the nonterminal label, the head word and its part of speech. The generation of fully-lexicalized modifying nonterminals is done in two steps, to allow for the parameters to be independently smoothed, which, in turn, is done to avoid sparse data problems. These two steps estimate the joint event of all three components using the chain rule. First, a *partially-lexicalized* version of the nonterminal is generated, consisting of the unlexicalized label plus the part of speech of its head word. These partially-lexicalized modifying nonterminals

¹⁶Our parsing engine allows an arbitrary mechanism for storage and discharge of requirements: they can be multisets, ordered lists, integers (simply to constrain the number of requirements), or any other mechanism. The mechanism used is determined at run-time.

are generated conditioning on the parent label, the head label, the head word, the head tag, the current state of the dynamic subcat and a distance metric. Symbolically, the parameter classes are

$$P_L(L(t)_i | \alpha(P), \gamma(H), w_h, t_h, subcat_L, \Delta_L) \quad (3.7)$$

$$P_R(R(t)_i | \alpha(P), \gamma(H), w_h, t_h, subcat_R, \Delta_R) \quad (3.8)$$

where Δ denotes the distance metric.¹⁷ As discussed above, one of the two components of this distance metric is the *vi* (“verb intervening”) predicate. The other is a predicate that simply reports whether the current modifier is the first modifier being generated, that is, whether $i = 1$. The second step is to generate the head word itself, where, because of the chain rule, the conditioning context consists of everything in the histories of equations (3.7) and (3.8) plus the partially-lexicalized modifier. As there are some interesting idiosyncrasies with these head-word-generation parameters, we describe them in more detail in §3.5.8.

3.5.4 The punctuation and coordinating conjunction parameter classes

3.5.4.1 Inconsistent model

As mentioned in §3.3.8, punctuation is raised to the highest position in the tree. This means that, in some sense, punctuation acts very much like a coordinating conjunction, in that it “conjoints” the two siblings between which it sits. Observing that it might be helpful for conjunctions to be generated conditioning on both of their conjuncts, Collins introduced two new parameter classes in his thesis parser, P_{punc} and P_{CC} .¹⁸

As per the definition of a conjoined phrase in §3.3.1, conjunction via a CC node or a punctuation node always occurs post-head (*i.e.*, as a right sibling of the head). Put another way, if a conjunction or punctuation mark occurs pre-head, it is not generated via this mechanism.¹⁹ Furthermore, even

¹⁷Throughout this thesis we will use the notation $L(w, t)_i$ to refer to the three items that comprise a fully-lexicalized left-modifying nonterminal, which are the unlexicalized label L_i , its head word w_{L_i} and its part of speech t_{L_i} , and similarly for right modifiers. We use $L(t)_i$ to refer to the two items L_i and t_{L_i} of a partially-lexicalized nonterminal. Finally, when we do not wish to distinguish between a left and right modifier, we use $M(w, t)_i$, $M(t)_i$ and M_i .

¹⁸Collins’ thesis does not say what the back-off structure of these new parameter classes is, *i.e.*, how they should be smoothed. We have included this information in the complete smoothing table in the Appendix.

¹⁹In fact, if punctuation occurs before the head, it is not generated at all—a deficiency in the parsing model that appears to be a holdover from the deficient punctuation handling in the model of (Collins, 1997).

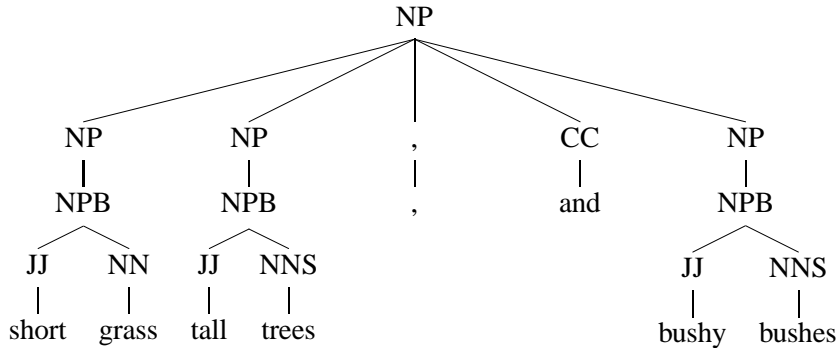


Figure 3.9: A tree containing both punctuation and conjunction.

if there is arbitrary material between the right conjunct and the head, the parameters effectively assume that the left conjunct is always the head child. For example, in Figure 3.9, the rightmost NP (“bushy bushes”) is considered to be conjoined to the leftmost NP (“short grass”), which is the head child, even though there is an intervening NP (“tall trees”).

The new parameters are incorporated into the model by requiring that *all* modifying nonterminals be generated with two boolean flags: `coord`, indicating that the nonterminal is conjoined to the head via a CC, and `punc`, indicating that the nonterminal is conjoined to the head via a punctuation mark. When either or both of these flags is true, the intervening punctuation or conjunction is generated via appropriate instances of the P_{punc}/P_{CC} parameter classes.

For example, the model generates the five children in Figure 3.9 in the following order: first, the head child is generated, which is the leftmost NP (“short grass”), conditioning on the parent label and the head word and tag. Then, since modifiers are always generated from the head outward, the right sibling of the head, which is the “tall trees” NP, is generated with both the `punc` and `CC` flags `false`. Then, the rightmost NP (“bushy bushes”) is generated with both the `punc` and `CC` booleans `true`, since it is considered to be conjoined to the head child and requires the generation of an intervening punctuation mark and conjunction. Finally, the intervening punctuation is generated conditioning on the parent, the head *and* the right conjunct, including the head words of the two conjoined phrases, and the intervening CC is similarly generated. A simplified version of the

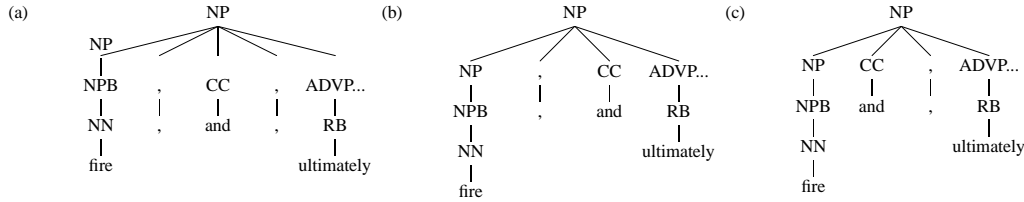


Figure 3.10: The Collins model assigns equal probability to these three trees.

probability of generating all these children is summarized as follows:

$$\begin{aligned}
 & \hat{p}_H(\text{NP} \mid \text{NP}, \text{grass}, \text{NN}) \cdot \\
 & \hat{p}_R(\text{NP}(\text{trees}, \text{NNS}), \text{punc} = \emptyset, \text{coord} = \emptyset \mid \text{NP}, \text{NP}, \text{grass}, \text{NN}) \cdot \\
 & \hat{p}_R(\text{NP}(\text{bushes}, \text{NNS}), \text{punc} = 1, \text{coord} = 1 \mid \text{NP}, \text{NP}, \text{grass}, \text{NN}) \cdot \\
 & \hat{p}_{\text{punc}}(, (,) \mid \text{NP}, \text{NP}, \text{NP}, \text{bushes}, \text{NNS}, \text{grass}, \text{NN}) \cdot \\
 & \hat{p}_{\text{CC}}(\text{CC}(\text{and}) \mid \text{NP}, \text{NP}, \text{NP}, \text{bushes}, \text{NNS}, \text{grass}, \text{NN})
 \end{aligned} \tag{3.9}$$

The idea is that, using the chain rule, the generation of two conjuncts and that which conjoins them is estimated as one, large joint event.²⁰

This scheme of using flags to trigger the $P_{\text{punc}}/P_{\text{CC}}$ parameters is problematic, at least from a theoretical standpoint, as it causes the model to be inconsistent. Figure 3.10 shows three different trees that would all receive the same probability from Collins’ model. The problem is that coordinating conjunctions and punctuation are not generated as first-class words, but *only* as triggered from these *punc* and *coord* flags, meaning that the number of such intervening conjunctive items (and the order in which they are to be generated) is not specified. So, for a given sentence/tree pair containing a conjunction and/or a punctuation mark, there is an infinite number of similar sentence/tree pairs with arbitrary amounts of “conjunctive” material between the same two nodes. Because all of these trees have the same, non-zero probability, the sum $\sum_T P(T)$ diverges, where T is a possible tree generated by the model, meaning the model is inconsistent (Booth and Thompson, 1973). Another consequence of not generating post-head conjunctions and punctuation as first-class words is that they do not count when calculating the head-adjacency component of

²⁰In (3.9), for clarity we have left out subcat generation and the use of Collins’ distance metric in the conditioning contexts. We have also glossed over the fact that lexicalized modifying nonterminals are actually generated in two steps, using two differently-smoothed parameters.

Collins’ distance metric.

When emulating Collins’ model, instead of reproducing the P_{punc} and P_{CC} parameter classes directly in our parsing engine, we chose to use a different mechanism that does not yield an inconsistent model, but still estimates the large joint event that was the motivation behind these parameters in the first place.

3.5.4.2 History mechanism

In our emulation of Collins’ model, we use the *history* to estimate the joint event of generating a conjunction (or punctuation mark) and its two conjuncts. The first big change is that we treat punctuation preterminals and CCs as first-class objects, meaning that they are generated just like any other modifying nonterminal. The second change is a little more involved.

First, we redefine the distance metric to consist solely of the *vi* predicate. Then, we add to the conditioning context a mapped version of the previously-generated modifier according to the following mapping function:

$$\delta(M_i) = \begin{cases} +START+ & \text{if } i = 0, \\ CC & \text{if } M_i = CC, \\ +PUNC+ & \text{if } M_i = , \text{ or } M_i = :, \\ +OTHER+ & \text{otherwise.} \end{cases} \quad (3.10)$$

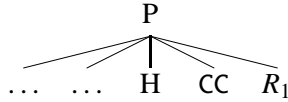
where M_i is some modifier L_i or R_i .²¹ So, our the maximal context for our modifying nonterminal parameter class is now defined as follows:

$$P_M(M(t)_i | \alpha(P), \gamma(H), w_h, t_h, subcat_{side}, vi(M_i), \delta(M_{i-1}), side) \quad (3.11)$$

where *side* is a boolean-valued event that indicates whether the modifier is on the left or right side of the head. By treating CC and punctuation nodes as first-class nonterminals and by adding the

²¹Originally, we also had an additional mechanism that attempted to generate punctuation and conjunctions with conditional independence. One of our reviewers astutely pointed out that the mechanism led to a deficient model (the very thing we have been trying to avoid), and so we have subsequently removed it from our model. The removal leads to a 0.05% absolute reduction in F-measure (which in this case is also a 0.05% relative increase in error) on sentences of length ≤ 40 words in §00 of the Penn Treebank. As this difference is not at all statistically significant (according to a randomized stratified shuffling test (Cohen, 1995)), all evaluations reported in this chapter are with the original model.

mapped version of the previously-generated modifier, we have, in one fell swoop, incorporated the “no intervening” component of Collins’ distance metric (the $i = 0$ case of the δ function) and achieved an estimate of the joint event of a conjunction and its conjuncts, albeit with different dependencies, *i.e.*, a different application of the chain rule. To put this parameterization change in sharp relief, consider the abstract tree structure



To a first approximation, under the old parameterization, the conjunction of some node R_1 with a head H and a parent P looked like

$$\hat{p}_H(H | P) \cdot \hat{p}_R(R_1, \text{coord} = 1 | P, H) \cdot \hat{p}_{CC}(CC | P, H, R_1),$$

whereas under the new parameterization, it looks like

$$\hat{p}_H(H | P) \cdot \hat{p}_R(CC | P, H, +\text{START}+) \cdot \hat{p}_R(R_1 | P, H, CC).$$

Either way, the probability of the joint conditional event $\{H, CC, R_1 | P\}$ is being estimated, but with the new method, there is no need to add two, new specialized parameter classes, and it does not introduce inconsistency into the model. Using less simplification, the probability of generating the five children of Figure 3.9 is now

$$\begin{aligned} & \hat{p}_H(\text{NP} | \text{NP}, \text{grass}, \text{NN}) \cdot \\ & \hat{p}_M(\text{NP}(\text{trees}, \text{NNS}) | \text{NP}, \text{NP}, \text{grass}, \text{NN}, \{\}, \text{false}, +\text{START}+, \text{right}) \cdot \\ & \hat{p}_M(, (, ,) | \text{NP}, \text{NP}, \text{grass}, \text{NN}, \{\}, \text{false}, +\text{OTHER}+, \text{right}) \cdot \\ & \hat{p}_M(\text{CC}(\text{and}, \text{CC}) | \text{NP}, \text{NP}, \text{grass}, \text{NN}, \{\}, \text{false}, +\text{PUNC}+, \text{right}) \cdot \\ & \hat{p}_M(\text{NP}(\text{bushes}, \text{NNS}) | \text{NP}, \text{NP}, \text{grass}, \text{NN}, \{\}, \text{false}, \text{CC}, \text{right}) \end{aligned} \quad (3.12)$$

As shown in §3.7, this new parameterization yields virtually identical performance to that of the Collins model.²²

²²As described in Chapter 4, our parsing engine allows easy experimentation with a wide variety of different generative models, including the ability to construct history contexts from arbitrary numbers of previously-generated modifiers. The mapping function δ and the transition function τ presented in this section are just two examples of this capability.

3.5.5 The base NP model: a model unto itself

As we have already seen, there are several ways in which base NPs are exceptional in Collins’ parsing model. This is partly because the flat structure of base NPs in the Penn Treebank suggested the use of a completely different model by which to generate them. Essentially, the model for generating children of NPB nodes is a “bigrams of nonterminals” model. That is, it looks a great deal like a bigram language model, except that the items being generated are not words, but lexicalized nonterminals. Heads of NPB nodes are generated using the normal head-generation parameter, but modifiers are generated always conditioning not on the head, but on the previously-generated modifier. That is, we modify equations (3.7) and (3.8) to be

$$P_{L,NPB}(L(t)_i | P, L(w, t)_{i-1}) \quad (3.13)$$

$$P_{R,NPB}(R(t)_i | P, R(w, t)_{i-1}) \quad (3.14)$$

Though it is not entirely spelled out in his thesis, Collins considers the previously-generated modifier to *be* the head child, for all intents and purposes. Thus, the subcat and distance metrics are always irrelevant, since it is as though the current modifier is right next to the head.²³ Another consequence of this is that NPBs are never considered to be coordinated phrases (as mentioned in §3.3.12), and thus CCs dominated by NPB are never generated using a P_{CC} parameter; instead, they are generated using a normal modifying-nonterminal parameter. Punctuation dominated by NPB, on the other hand, is still, as always, generated via P_{punc} parameters, but where crucially, the modifier is always conjoined (via the punctuation mark) to the “pseudo-head” that is the previously-generated modifier. Consequently, when generating some right modifier R_i , the previously-generated modifier on the right side of the head, R_{i-1} , is never a punctuation preterminal, but always the previous “real” (*i.e.*, non-punctuation) preterminal.²⁴

Base NPs are also exceptional with respect to determining chart item equality, the comma pruning rule and general beam-pruning (see §3.6.2 for details).

²³This is the main reason that the *cv* (“contains verb”) predicate is always *false* for NPBs, as that predicate only applies to material that intervenes between the current modifier and the head.

²⁴Interestingly, unlike the regular model, punctuation that occurs to the left of the head *is* generated when it occurs within an NPB. Thus, this particular—albeit small—deficiency of Collins’ punctuation-handling does not apply to the base NP model.

3.5.6 Parameter classes for priors on lexicalized nonterminals

Two parameter classes that only make their appearance in Appendix E of Collins’ thesis are those that compute maximum-likelihood marginals for lexicalized nonterminals. These marginals are used as a crude proxy for the outside probability of a chart item (see (Baker, 1979; Lari and Young, 1990) for full descriptions of the Inside–Outside algorithm). Previous work (Goodman, 1997a) has shown that the inside probability alone is an insufficient scoring metric when comparing chart items covering the same span during decoding, and that some estimate of the outside probability of a chart item should be factored into the score. A marginal on the root (lexicalized) nonterminal label of the derivation forest represented by a particular chart item is used for this purpose in Collins’ parser.

The marginal of a lexicalized nonterminal $M(w, t)$ is broken down into two, separate estimates using parameters from two new classes, P_{prior_w} and $P_{prior_{NT}}$:

$$P_{prior}(M(w, t)) = P_{prior_w}(w, t) \cdot P_{prior_{NT}}(M | w, t)$$

where $\hat{p}(M | w, t)$ is smoothed with $\hat{p}(M | t)$, and where estimates using the parameters of the P_{prior_w} class are unsmoothed.

3.5.7 Smoothing weights

Many of the parameter classes in Collins’ model—and indeed, in most statistical parsing models—define conditional probabilities with very large conditioning contexts. In this case, the conditioning contexts represent some subset of the *history* of the generative process. Even if there were orders of magnitude more training data available, the large size of these contexts would cause horrendous sparse data problems. The solution is to *smooth* these distributions that are made rough primarily by the abundance of zeros. Collins uses the technique of *deleted interpolation*, which smooths the distributions based on full contexts with coarser models that use less of the context, by successively deleting elements from the context at each back-off level. As a simple example, the head parameter class smooths $P_{H_0}(H | P, w_h, t_h)$ with $P_{H_1}(H | P, t_h)$ and $P_{H_2}(H | P)$. For some conditional probability $p(A | B)$, let us call the reduced context at the i th back-off level $\phi_i(B)$, where typically

$\phi_0(B) = B$.²⁵ Each estimate in the back-off chain is computed via maximum likelihood estimation, and the overall smoothed estimate is computed using $n - 1$ smoothing weights for n back-off levels, denoted $\lambda_0, \dots, \lambda_{n-2}$. These weights are used in a recursive fashion: the smoothed version $\tilde{e}_i = \tilde{p}_i(A | \phi_i(B))$ of an unsmoothed ML estimate $e_i = \hat{p}_i(A | \phi_i(B))$ at back-off level i is computed via the formula

$$\tilde{e}_i = \lambda_i e_i + (1 - \lambda_i) \tilde{e}_{i+1}, \quad 0 \leq i < n - 1, \quad \tilde{e}_{n-1} = e_{n-1}. \quad (3.15)$$

So, for example, with three levels of back-off, the overall smoothed estimate would be defined as

$$\tilde{e}_0 = \lambda_0 e_0 + (1 - \lambda_0) [\lambda_1 e_1 + (1 - \lambda_1) e_2] \quad (3.16)$$

It is easy to prove by structural induction that if

$$0 \leq \lambda_i \leq 1 \text{ and } \sum_A \hat{p}_i(A | \phi_i(B)) = 1, \quad 0 \leq i < n - 1,$$

then

$$\sum_A \tilde{p}_0(A | \phi_0(B)) = 1. \quad (3.17)$$

Each smoothing weight can be conceptualized as the confidence in the estimate with which it is being multiplied. These confidence values can be derived in a number of sensible ways; the technique used by Collins was adapted from that used in (Bikel et al., 1997), which makes use of a quantity called the *diversity* of the history context (Witten and Bell, 1991), which is equal to the number of unique futures observed in training for that history context.

3.5.7.1 Deficient model

As mentioned, n back-off levels require $n - 1$ smoothing weights. Collins' parser effectively uses n weights, because the estimator always adds an extra, constant-valued estimate to the back-off chain. Collins' parser hard-codes this extra value to be a vanishingly-small (but non-zero) “probability” of 10^{-19} , resulting in smoothed estimates of the form

$$\tilde{e}_0 = \lambda_0 e_0 + (1 - \lambda_0) \left[\lambda_1 e_1 + (1 - \lambda_1) \left[\lambda_2 e_2 + (1 - \lambda_2) \cdot 10^{-19} \right] \right] \quad (3.18)$$

²⁵In a more general version of deleted interpolation, the suite of history-mapping functions $\phi_0, \dots, \phi_{n-1}$ merely need to have the property $\phi_i(B) \subseteq \phi_{i+1}(B)$ for $0 \leq i < n - 1$ and for all B . Having mapping functions that successively delete more elements from some complex history context B is a special case of this property. In our parsing engine, we only require the looser condition.

when there are three levels of back-off. The addition of this constant-valued $e_n = 10^{-19}$ causes all estimates in the parser to be deficient, as it ends up throwing away probability mass. More formally, the proof leading to Equation (3.17) no longer holds: the “distribution” sums to less than one (there is no history context in the model for which there are 10^{19} possible outcomes).²⁶

3.5.7.2 Smoothing factors and smoothing terms

The formula given in Collins’ thesis for computing smoothing weights is

$$\lambda_i = \frac{c_i}{c_i + 5u_i}$$

where c_i is the count of the history context $\phi_i(B)$ and u_i is the diversity of that context.²⁷ The multiplicative constant 5 is used to decrease all smoothing values, effectively giving less weight to the back-off levels with more context. The value of this constant was optimized by looking at overall parsing performance on the development test set, Section 00 of the Penn Treebank. We call this constant the *smoothing factor*, and denote it as f_f . As it happens, the actual formula for computing smoothing weights in Collins’ implementation is

$$\lambda_i = \begin{cases} \frac{c_i}{c_i + f_t + f_f u_i} & \text{if } c_i > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (3.19)$$

where f_t is an unmentioned *smoothing term*. For every parameter class except the subcat parameter class and P_{prior_w} , $f_t = 0$ and $f_f = 5.0$. For the subcat parameter class, $f_t = 5.0$ and $f_f = 0$. For

²⁶Collins used this technique to ensure that even futures that were never seen with an observed history context would still have some probability mass, albeit a vanishingly small one (Collins, p.c.). Another commonly-used technique would be to back-off to the uniform distribution, which has the desirable property of not producing deficient estimates. As with all of the Treebank- or model-specific aspects of the Collins parser, our engine uses Equation (3.16) or (3.18) depending on the value of a run-time setting.

²⁷The smoothing weights can be viewed as confidence values for the probability estimates with which they are multiplied. The Witten-Bell technique crucially makes use of the quantity $\bar{n}_i = \frac{c_i}{u_i}$, the average number of transitions from the history context $\phi_i(B)$ to a possible future. With a little algebraic manipulation, we have

$$\lambda_i = \frac{\bar{n}_i}{\bar{n}_i + 5},$$

a quantity that is at its maximum when $\bar{n}_i = c_i$ and at its minimum when $\bar{n}_i = 1$, that is, when every future observed in training was unique. This latter case represents when the model is most “uncertain”, in that the transition distribution from $\phi_i(B)$ is uniform and poorly-trained (1 observation per possible transition). Because these smoothing weights measure, in some sense, the closeness of the observed distribution to uniform, they can be viewed as proxies for the entropy of the distribution $p(\cdot | \phi_i(B))$.

Back-off level	$P_{L_w}(w_{L_i} \dots)$ $P_{R_w}(w_{R_i} \dots)$
0	$\gamma(L_i), t_{L_i}, \text{coord, punc, } \alpha(P), \gamma(H), w_h, t_h, \Delta_L, \text{subcat}$
1	$\gamma(L_i), t_{L_i}, \text{coord, punc, } \alpha(P), \gamma(H), t_h, \Delta_L, \text{subcat}$
2	t_{L_i}

Table 3.1: Back-off levels for P_{L_w}/P_{R_w} , the modifier head-word generation parameter classes. w_{L_i} and t_{L_i} are respectively the head word and its part of speech of the nonterminal L_i . This table is basically a reproduction of the last column of Table 7.1 in Collins’ thesis.

$P_{prior_w}, f_t = 1.0$ and $f_f = 0.0$. This curiously means that diversity is not used at all when smoothing subcat-generation probabilities.²⁸

The second case in (3.19) handles the situation where the history context was never observed in training, *i.e.*, where $c_i = u_i = 0$, which would yield an undefined value when $f_t = 0$. In such situations, by making $\lambda_i = 0$, all remaining probability mass gets thrown to the smoothed back-off estimate, \tilde{e}_{i+1} . This is a crucial part of the way smoothing is done: if a particular history context $\phi_i(B)$ has *never* been observed in training, the smoothed estimate using less context, $\phi_{i+1}(B)$, is simply substituted as the “best guess” for the estimate using more context; that is, $\tilde{e}_i = \tilde{e}_{i+1}$.²⁹

3.5.8 Modifier head-word generation

As mentioned above in §3.5.3, fully-lexicalized modifying nonterminals are generated in two steps. First, the label and part-of-speech tag are generated with an instance of P_L or P_R . Next, the head word is generated via an instance of one of two parameter classes, P_{L_w} or P_{R_w} . The back-off contexts for the smoothed estimates of these parameters is specified in Table 3.1. Notice how the last level of back-off is markedly different from the previous two levels, in that it removes nearly *all* the elements of the history: in the face of sparse data, the probability of generating the head word of a modifying nonterminal is conditioned only on its part of speech.

²⁸As mentioned above, the P_{prior_w} parameters are unsmoothed. However, due to the deficient estimation method, they still have an associated lambda value, the computation of which, just like the subcat-generation probability estimates, does not make use of diversity.

²⁹This fact will be crucial toward understanding how little the Collins parsing model relies on bilexical statistics, as described in §6.1 and the supporting experiment shown in Table 6.1.

Back-off level	$P_{M_w}(w_{M_i} \dots)$
0	$\gamma(M_i), t_{M_i}, \alpha(P), \gamma(H), w_h, t_h, \text{subcat}_{side}, \mathbf{vi}(M_i), \delta(M_{i-1}), \text{side}$
1	$\gamma(M_i), t_{M_i}, \alpha(P), \gamma(H), t_h, \text{subcat}_{side}, \mathbf{vi}(M_i), \delta(M_{i-1}), \text{side}$
2	t_{M_i}

Table 3.2: Our new parameter class for the generation of head words of modifying nonterminals.

3.5.8.1 Smoothing and the last level of back-off

The table of Table 3.1 is misleading, however. In order to capture the most data for the crucial last-level of back-off, Collins uses words that occur on *either side* of the head word, resulting in a general estimate $\hat{p}(w|t)$, as opposed to $\hat{p}_{L_w}(w_{L_i} | t_{L_i})$. Accordingly, in our emulation of Collins model, we replace the left- and right-word parameter classes with a single modifier head-word generation parameter class that, as with (3.11), includes a boolean *side* component that is deleted from the last level of back-off. See Table 3.2.

Even with this change, there is still a problem. Every head word in a lexicalized parse tree is the modifier of some other head word—*except* the word that is the head of the entire sentence (*i.e.*, the head word of the root nonterminal). In order to properly duplicate Collins’ model, an implementation must take care that the $P(w|t)$ model includes counts for these important head words.³⁰

3.5.8.2 Unknown word mapping

As mentioned above, instead of mapping every low-frequency word in the training data to some special +UNKNOWN+ token, Collins’ trainer instead leaves the training data untouched, and instead selectively maps words that appear in the back-off levels of the parameters from the P_{L_w} and P_{R_w} parameter classes. Rather curiously, the trainer only maps words that appear in the *futures* of these parameters, but never in the histories. Put another way, low-frequency words are generated as +UNKNOWN+, but are left unchanged when they are conditioned upon. For example, in Figure 3.11, where we assume “Fido” is a low-frequency word, the trainer would derive counts for the smoothed

³⁰In our implementation, we add such counts by having our trainer generate a “fake” modifier event where the observed, lexicalized root nonterminal is considered a modifier of +TOP+, the hidden nonterminal that is the parent of the observed root of every tree (see §3.5.9 for details on the +TOP+ nonterminal).

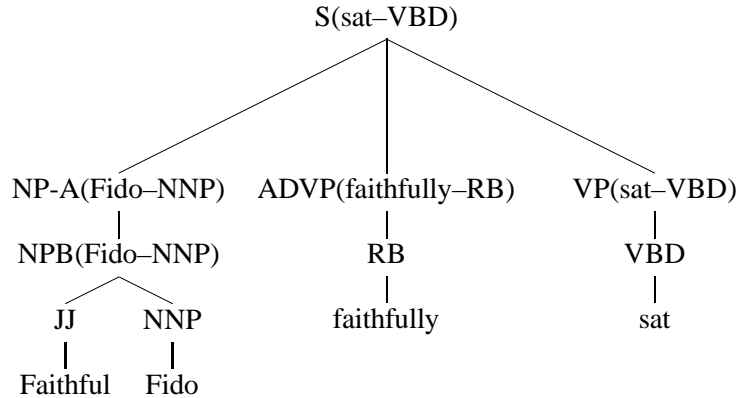


Figure 3.11: The low-frequency word “Fido” is mapped to +UNKNOWN+, but only when it is generated, not when it is conditioned upon. All the nonterminals have been lexicalized (except for preterminals) to show where the heads are.

parameter

$$p_{L_w} (+UNKNOWN + | NP-A, NNP, coord = 0, punc = 0, S, VP, sat, VBD, \dots) \quad (3.20)$$

However, when collecting events that condition on “Fido”, the word would not be mapped, such as the parameters

$$p_L (JJ(JJ) | NPB, NNP, Fido)$$

$$p_{L_w} (Faithful | JJ, JJ, NPB, NNP, Fido)$$

This strange mapping scheme has some interesting consequences. First, imagine what happens to words that are truly unknown, that *never* occurred in the training data. Such words are mapped to the +UNKNOWN+ token outright before parsing. Whenever the parser estimates a probability with this truly unknown word in the history, it will necessarily throw all probability mass to the backed-off estimate ($\tilde{\epsilon}_1$, in our earlier notation), since +UNKNOWN+ effectively *never* occurred in a history context during training.

The second consequence is that the mapping scheme yields a “superficient”³¹ model, if all other parts of the model are probabilistically sound (which is actually not the case here). With a parsing model such as Collins’ that uses bilinear dependencies, generating words in the course of

³¹The term “deficient” is used to denote a model in which one or more estimated distributions sums to less than 1. We use the term “superficient” to denote a model in which one or more estimated distributions sums to greater than 1.

parsing is very much like a bigram language model: every word is generated conditioning on some previously generated word, as well as some hidden material. The only difference is that the word being conditioned upon is often not the immediately-preceding word in the sentence. However, one could plausibly construct a consistent bigram language model that generates words with the same dependencies as are in a statistical parser that uses bilexical dependencies derived from head-lexicalization.

Collins (p.c.) notes that his parser’s unknown-word–mapping scheme can be made consistent if one were to add a parameter class that estimates $\hat{p}(w | +\text{UNKNOWN}+)$, where $w \in V_L \cup \{+\text{UNKNOWN}+\}$. The values of these estimates for a given sentence would be constant across all parses, meaning that the “superficiency” of the model would be irrelevant when determining $\arg \max_T P(T | S)$.

Proof of superficiality We prove the superficiality of Collins’ model due to the unknown word–mapping scheme by analogy with a standard bigram language model. The proof is provided here for completeness, and may be safely skipped.

Proof. Let V_o be the set of words observed during training *including low-frequency words*, and let $V = V_o \cup \{+\text{UNKNOWN}+\}$. Let $c : V_o \rightarrow \mathbb{Z}^+$ be a function that returns the count of each word in the training corpus, and let t be the unknown word threshold. Let ψ be the Collins-style unknown word–mapping function defined as follows:

$$\psi(w) = \begin{cases} +\text{UNKNOWN}+ & \text{if } c(w) < t, \\ w & \text{otherwise.} \end{cases}$$

For all sequences of words $S \in V^* = w_1 w_2 \cdots w_n$, $n \geq 0$, let M be a bigram language model that uses ψ as it is used in Collins’ parser. Now consider that, if this language model is consistent, it has the property that

$$\begin{aligned} \sum_S P_M(S) &= \sum_{n=0}^{\infty} \sum_{w_1} \cdots \sum_{w_n} p(\psi(w_1) | +\text{START}+) \cdots p(\psi(w_n) | w_{n-1}) p(+\text{STOP}+ | w_n) \\ &= 1 \end{aligned}$$

where the $n = 0$ case is simply $p(+\text{STOP}+ | +\text{START}+)$, that is, the probability of generating the empty string. Let V_L be the set of low-frequency words and let V_K be the set of non–low-frequency

words. More formally, let

$$V_L = \{w \in V_o : c(w) < t\}$$

$$V_K = \{w \in V_o : c(w) \geq t\}$$

Note that $V = V_L \cup V_K \cup \{+\text{UNKNOWN}+\}$. Now, if the parameters of M have been trained in an analogous way to Collins' trainer's bilexical parameters, then

$$\forall w_{i-1} \in V : \hat{p}(+\text{UNKNOWN}+ | w_{i-1}) + \sum_{w_i \in V_K} \hat{p}(w_i | w_{i-1}) = 1. \quad (3.21)$$

That is, for all words that can appear in the history, the parameters estimate a proper distribution for the *subset* of V that can appear in the future (which is $V - V_L$), meaning that the observed distribution sums to 1. Let $w_{i-1} \in V$ be a word for which $\hat{p}(+\text{UNKNOWN}+ | w_{i-1}) > 0$. If $V_L \neq \emptyset$, it is then the case that

$$\begin{aligned} \sum_{w_i \in V} \hat{p}(\psi(w_i) | w_{i-1}) &= (|V_L| + 1) \cdot \hat{p}(+\text{UNKNOWN}+ | w_{i-1}) + \sum_{w_i \in V_K} \hat{p}(w_i | w_{i-1}) \\ &> 1. \end{aligned} \quad (3.22)$$

As follows from the definition of consistency of a regular stochastic grammar (Thomason, 1986, pages 126–128), Equation 3.22 implies that

$$\sum_{\mathbf{W}} P_M(\mathbf{W}) > 1. \quad \square$$

3.5.9 The top parameter classes

It is assumed that all trees that can be generated by the model have an implicit nonterminal +TOP+ that is the parent of the observed root. The observed, lexicalized root nonterminal is generated conditioning on +TOP+ (which has a prior probability of 1.0) using a parameter from the class P_{TOP} . This special parameter class is mentioned in a footnote in Chapter 7 of Collins' thesis. There are actually two parameter classes used to generate observed roots, one for generating the partially-lexicalized root nonterminal, the other for generating the head word of the entire sentence, which we will call $P_{TOP_{NT}}$ and P_{TOP_w} , respectively. Table 3.3 gives the unpublished back-off structure of these two additional parameter classes.

Back-off level	$P_{TOP_{NT}}(H(t) \dots)$	$P_{TOP_w}(w \dots)$
0	+TOP+	$t, H, +TOP+$
1	n/a	t

Table 3.3: Back-off structure for $P_{TOP_{NT}}$ and P_{TOP_w} , which estimate the probability of generating $H(w, t)$ as the root nonterminal of a parse tree. $P_{TOP_{NT}}$ is unsmoothed.

Note that P_{TOP_w} backs off to simply estimating $\hat{p}(w|t)$. Technically, it should be estimating $\hat{p}_{NT}(w|t)$, which is to say the probability of a word occurring with a tag in the space of lexicalized nonterminals. This is different from the last level of back-off in the modifier head-word parameter classes, which is effectively estimating $\hat{p}(w|t)$ in the space of lexicalized preterminals. The difference is that in the same sentence, the same head word can occur with the same tag in multiple nodes, such as “sat” occurring with the tag VBD three times (instead of just once) in the tree shown in Figure 3.8. Despite this difference, Collins’ parser uses counts from the (shared) last level of back-off of the P_{L_w}/P_{R_w} parameters when delivering e_1 estimates for the P_{TOP_w} parameters. Our parsing engine emulates this “count sharing” for P_{TOP_w} by default, by sharing counts from our P_{M_w} parameter class.

3.6 Decoding

Parsing, or decoding, is performed via a probabilistic version of the CKY chart-parsing algorithm. As with normal CKY, even though the model is defined in a top-down, generative manner, decoding proceeds bottom-up. Collins’ thesis gives a pseudocode version of his algorithm in an appendix. This section contains a few practical details.

3.6.1 Chart item equivalence

Since the goal of the decoding process is to determine the maximally-likely theory, if during decoding a proposed chart item is equivalent to an item that is already in the chart, the one with the greater score survives. Chart item equivalence is closely tied to the generative parameters used to construct derivations: we want to treat two chart items as inequivalent if they represent derivation forests that would be considered inequivalent according to the output elements and conditioning

contexts of the parameters used to generate them, subject to the independence assumptions of the model. For example, for two chart items to be considered equivalent, they must have the same label (the label of the root of their respective derivation forests' subtrees), the same head word and tag and the same left and right subcat. They must also have the same head label (that is, label of the head child).

If a chart item's root label is an NP node, its head label is most often an NPB node, given the "extra" NP levels that are added during preprocessing to ensure that NPB nodes are always dominated by NP nodes. In such cases, the chart item will contain a back-pointer to the chart item that represents the base NP. Curiously, however, Collins' implementation considers the head label of the NP chart item not to be NPB, but rather the head label of the NPB chart item. In other words, to get the head label of an NP chart item, one must "peek through" the NPB and get at the NPB's head label. Presumably, this was done as a consideration for the NPB nodes being "extra" nodes, in some sense. It appears to have little effect on overall parsing accuracy, however.

3.6.2 Pruning

Ideally, every item could be kept in the chart, and when the root symbol has been generated for all partial derivations, the top-ranked one would "win". In order to speed things up, Collins employs three different types of pruning. The first form of pruning is to use a beam: the chart memoizes the highest-scoring derivation in each span, and if a proposed chart item for that span is not within a certain factor of the top-scoring item, it is not added to the chart. Collins reports in his thesis that he uses a beam width of 10^5 . As it happens, the beam width for his thesis experiments was 10^4 . Interestingly, there is a negligible difference in overall parsing accuracy when this wider beam is used (see §3.7, Table 3.5). An interesting modification to the standard beam in Collins' parser is that for chart items representing NP or NP-A derivations with more than one child, the beam is expanded to be $10^4 \cdot e^3$. We suspect that Collins made this modification to handle the greater perplexity associated with NPs after he added the base NP model.

The second form of pruning employed is using a comma constraint. Collins observed that in the Penn Treebank data, 96% of the time, when a constituent contained a comma, the word

immediately following the end of the constituent's span was either a comma or the end of the sentence. So, for speed reasons, the decoder rejects all derivations that would generate constituents that violate this comma constraint.³² There is a subtlety to Collins' implementation of this form of pruning, however. Commas are quite common within parenthetical phrases. Accordingly, if a comma in an input sentence occurs after an open parenthesis and before a closing parenthesis or the end of the sentence, it is not considered a comma for the purposes of the comma constraint. Another subtlety is that the comma constraint should effectively *not* be employed when pursuing derivations of an NPB subtree. As it turns out, using the comma constraint also affects accuracy, as shown in §3.7.

The final form of pruning employed is rather subtle: within each cell of the chart that contains items covering some span of the sentence, Collins' parser uses buckets of items that all share the same root nonterminal label for their respective derivations. Only 100 of the top-scoring items covering the same span with the same nonterminal label are kept in a bucket, meaning that if a new item is proposed and there are already 100 items covering the same span with the same label in the chart, then it will be compared to the lowest-scoring item in the bucket. If it has a higher score, it will be added to the bucket and the lowest-scoring item will be removed; otherwise, it will not be added. Apparently, this type of pruning has little effect, and so we have not duplicated it in our engine.³³

3.6.3 Unknown words and parts of speech

When the parser encounters an unknown word, the first-best tag delivered by Ratnaparkhi's tagger (Ratnaparkhi, 1996) is used. As it happens, the tag dictionary built up when training contains entries for every word observed, even if it was a low-frequency word. This means that when decoding, the output of the tagger is used only for those words that are truly unknown, *i.e.*, that were never observed in training. For all other words, the chart is seeded with a separate item for each tag

³²If one generates commas as first-class words, as we have done, one must take great care in applying this comma constraint, for otherwise, chart items that represent partially-completed constituents (*i.e.*, constituents for which not all modifiers have been generated) may be incorrectly rejected. This is especially important for NPB constituents.

³³Although we *have* implemented a version of this type of pruning that limits the number of items that can be collected in any one cell, that is, the maximum number of items that cover a particular span.

Model	Performance on Section 00					
	LR	LP	CBs	0 CBs	≤ 2 CBs	F
Collins' Model 2	89.75	90.19	0.77	69.10	88.31	89.97
Baseline (Model 2 emulation)	89.89	90.14	0.78	68.82	89.21	90.01
Clean-room Model 2	88.85	88.97	0.92	65.55	87.06	88.91

Table 3.4: Overall parsing results using only details found in (Collins, 1997; Collins, 1999). The first two lines show the results of Collins' parser and those of our parser in its "complete" emulation mode (*i.e.*, including unpublished details). All reported scores are for sentences of length ≤ 40 words. LR/LP are the primary scoring metrics, labeled precision and labeled recall, respectively. CBs is the average number of crossing brackets. 0 CBs and ≤ 2 CBs are the percentage of sentences with 0 and ≤ 2 crossing brackets, respectively. F (the "F-measure") is the evenly-weighted harmonic mean of precision and recall, or $\frac{LP \cdot LR}{\frac{1}{2}(LP+LR)}$.

observed with that word in training.

3.7 Evaluation: Effects of Unpublished Details

In this section we present the results of effectively doing a "clean-room" implementation of Collins' parsing model, that is, using only information available in (Collins, 1997; Collins, 1999), as shown in Table 3.4.

The clean-room model has an 10.6% increase in F-measure error when compared to Collins' parser, and an 11.0% increase in F-measure error compared to our engine in its complete emulation of Collins' Model 2. This is comparable to the increase in error seen when removing such published features as the verb intervening component of the distance metric, where F-measure error increases by 9.86%, or the subcat feature, the removal of which results in a 7.62% increase in F-measure error.³⁴ Therefore, while the collection of unpublished details presented in §§3.3, 3.4, 3.5 & 3.6 is disparate, *in toto* they are every bit as important to overall parsing performance as certain of the published features.

³⁴These F-measures and the differences between them were calculated from experiments presented in (Collins, 1999, page 201), which, unlike our reported numbers, were on all sentences, not just those of length ≤ 40 words. As Collins notes, removing *both* the distance metric and subcat features results in a gigantic drop in performance, since without both of these features, the model has no way to encode the fact that flatter structures should be avoided in several crucial cases, such as for PPs, which tend to prefer one argument to the right of their head children.

Model description	Performance on Section 00					
	LR	LP	CBs	0 CBs	≤ 2 CBs	F
Collins' Model 2	89.75	90.19	0.77	69.10	88.31	89.97
Baseline (Model 2 emulation)	89.89	90.14	0.78	68.82	89.21	90.01
Unknown word threshold = 5 and unknown words handled in uniform way (see §3.4.3)	89.94	90.22	0.77	68.99	89.27	90.08
No training trees skipped (see §3.4.2)	89.85	90.12	0.78	68.71	89.16	89.98
Beam width = 10^5 †	89.90	90.14	0.78	68.93	89.16	90.02
Non-deficient estimation (see §3.5.7.1)	89.75	90.00	0.80	68.82	88.88	89.87
No comma constraint (see §3.6.2)	89.52	89.80	0.84	68.09	88.20	89.66
No universal $p(w t)$ model‡	89.40	89.17	0.88	66.14	87.92	89.28
Clean-room Model 2	88.85	88.97	0.92	65.55	87.06	88.91

Table 3.5: Effects of independently removing/changing individual details on overall parsing performance. All reported scores are for sentences of length ≤ 40 words. †With beam width = 10^5 , processing time was 3.36 times longer than with standard beam (10^4). ‡No count sharing was performed for P_{TOP_w} (see §3.5.9), and $p(w|t)$ estimates were side-specific (see §3.5.8.1).

This does not mean that all details are equally important. Table 3.5 shows the effect of independently removing or changing certain of the more than 30 unpublished details on overall parsing performance.³⁵ Often, the detrimental effect is quite insignificant, even in the performance-obsessed world of statistical parsing, and occasionally, the effect is not even detrimental at all. That is why we do not claim importance of any single unpublished detail, but rather that of their totality, given that several of the unpublished details are, most likely, interacting. However, we note that certain individual details, such as the “universal $p(w|t)$ model”, *do* appear to have a much more marked effect on overall parsing accuracy than others.

³⁵As an anonymous reviewer of this work pointed out, the use of the comma constraint is a “published” detail. However, the specifics of how certain commas do not apply to the constraint is an “unpublished detail”, as mentioned in §3.6.2.

Chapter 4

Designing a new parsing engine

4.1 Motivation

As previously mentioned, ever since the widespread availability of the Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993), there have been numerous, statistical parsers developed for English, *e.g.* (Magerman, 1995; Collins, 1997; Miller et al., 1998; Charniak, 2000). To varying degrees, these parsers and others—while very successful at the tasks for which they were designed—had the following limitations:

- they had a fairly fixed probabilistic structure, which could only be changed by re-coding some significant portion of the program¹
- they had hard-coded features specific to English
- they had hard-coded features specific to the Penn Treebank
- they were designed only for a uniprocessor environment

Building on our work in (Bikel, 2000) and (Bikel and Chiang, 2000), we have developed a design for a head-driven, chart parsing engine that addresses all of the above limitations, and we present

¹Based on remarks by Charniak at the NAACL 2000 conference, the parser of (Charniak, 2000) does not appear to have this limitation, possessing some flexible mechanism to specify different probability structures.

this design here.² In particular, our design

- provides appropriate layers of abstraction and encapsulation for quickly extending the engine to different languages and/or Treebank annotation styles,
- has “plug-’n’-play” probability structures with flexible mechanisms for altering dependencies,
- has a flexible constrained parsing facility and
- is multi-threaded for use in a multiprocessor and/or multihost environment for throughput superior to traditional uniprocessing parsers.

While historically the motivation for the design of our engine arose from working with BBN’s parser (Miller et al., 1998) (which uses a simpler variant of Collins’ model), the details and experiments with Collins’ model discussed in Chapter 3 suggest the desiderata for a new parsing engine design. Throughout this chapter, we will attempt to highlight the areas of the design that were driven by properties of the baseline model explored in the previous chapter.

4.2 Background

4.2.1 Plug-’n’-play probability structures and sentence-level parallelism

In preparation for the work presented in (Bikel, 2000), we explored a variety of lexical probability structures involving WordNet synsets (Fellbaum, Grabowski, and Landes, 1998). To facilitate this experimentation, we developed a “plug-’n’-play” lexical probability structure architecture. The model we were extending was extracted from BBN’s SIFT system (Miller et al., 1998), which is a history-based model that is derived in large part from Model 2 of (Collins, 1997).

For the computation of probability estimates, the BBN parser had hard-coded construction of data objects to represent the future and history events, computing a maximum-likelihood estimate of $P(X|Y)$ by $\frac{c(X,Y)}{c(Y)}$, where $c()$ delivers the count of the specified event in the training data. The

²The implementation of this design is available by visiting the URL <http://www.cis.upenn.edu/~dbikel/#stat-parser>.

architecture of (Bikel, 2000) placed a layer of abstraction at the computation of the histories for the various back-off levels of the parser: in pseudocode, the construction of a history object for a particular back-off level became simply

```
history = probabilityStructure.get(backOffLevel, fullContext);
```

We also made an initial foray into parallel-processing by developing a multi-threaded *sentence server*, to provide parallelism at the sentence level in a cluster computing environment. This work paved the way for the significantly greater degree of parallelism in the architecture of our current parsing engine.

4.2.2 Language independence

For (Bikel and Chiang, 2000), we took two parsing models, BBN’s SIFT-derived parser and David Chiang’s stochastic TAG parser (Chiang, 2000) and adapted them to parse Chinese. The results were that the Chinese-adapted models performed with accuracy close to their English counterparts when trained on comparably-sized corpora and tested on their respective domains.³ The main difficulty of the project was “ferreting out” and replacing all traces of English-specific or Treebank-specific code in the parsers, such as replacing code dependent on the standard Roman character encoding and finding and abstracting away from language-specific word features and parts of speech. This project greatly influenced the language-independent design of our current, extensible architecture, which allows the user to specify at run-time a “language package” that contains all data and methods specific to a particular language and/or Treebank annotation style.

4.3 Design Overview

The parsing engine we have developed supports a wide range of head-driven parsing models, including that of BBN’s SIFT system, as well as both Models 2 and 3 of (Collins, 1997). Figure 4.1 gives a pictorial overview of the design.

³Our two comparably-sized corpora were: ~100k words of WSJ text from the (English) Penn Treebank and ~100k words of Xinhua newswire text from the Chinese Treebank. The stochastic TAG model performed at 77%/78% labeled precision/recall on the Xinhua test set, compared to 79%/80% on the English test set. See (Bikel and Chiang, 2000) for more details.

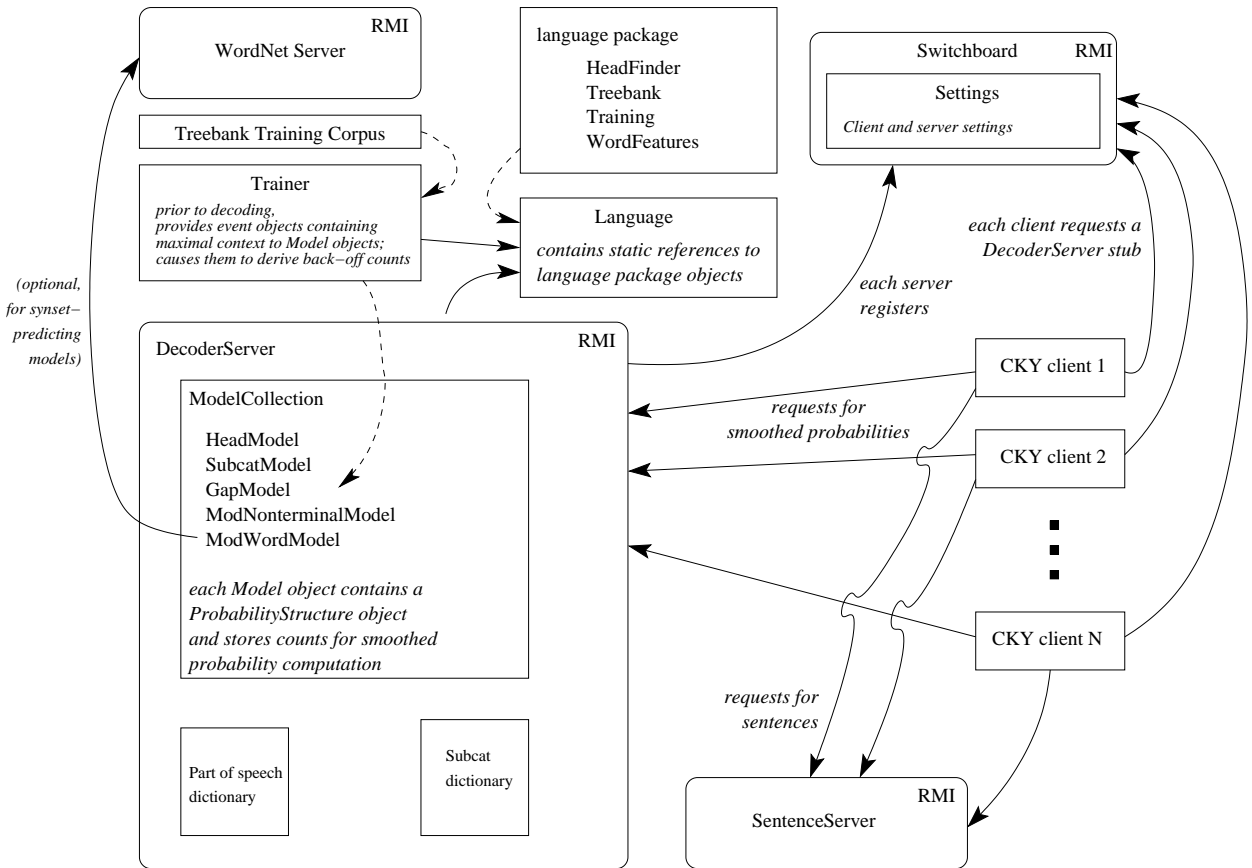


Figure 4.1: Pictorial design overview. Arrows indicate functional relationships. While there is necessarily always a bi-directional flow of information, the solid arrows indicate the direction of requests from a client to a server (or from a consumer to a producer). Dashed arrows explicitly indicate the flow of information or objects.

Control flow The `Switchboard` acts as the hub in a distributed-computing run. CKY decoder clients and `DecoderServer` modules are started on available hosts, either on the local multiprocessor or on hosts available via the network (typically the local area network). Both the decoders and the `DecoderServers` register themselves with the `Switchboard` upon startup. Next, each decoder requests a `DecoderServer` stub from the `Switchboard` (this “stub” allows the decoder to communicate directly with its `DecoderServer`). Finally, each decoder enters a loop where it requests an unprocessed sentence from the `Switchboard`. When all sentences of a file have been processed, the `Switchboard` assembles them in the order in which they appeared in the input file and creates an output file. When all files have been processed, the `Switchboard` tells all registered decoders and `DecoderServers` to die.

The engine was engineered to allow the loose coupling between decoders and the decoder server modules described above. However, when there is enough RAM per CPU to accommodate both the model and a decoder, it is more efficient to have each decoder start its own private `DecoderServer` with which it will communicate directly, instead of via a network communication protocol.

Fault tolerance If any decoder dies, processing will continue uninterrupted; if the decoder that died was in the midst of parsing a particular sentence, the `Switchboard` ensures that that sentence is handed off to another decoder the next time a sentence is requested. If any `DecoderServer` dies, all decoders that had been using that `DecoderServer` will automatically request a new `DecoderServer` from the `Switchboard`, which will comply by returning the stub of the most lightly-loaded `DecoderServer` for each request; in fact, a `DecoderServer` failure can happen right in the middle of parsing, and the replacement `DecoderServer` will be gotten and used without affecting the higher-level parsing process. The `Switchboard` keeps track of the decoders’ incremental work via a log file, so that even if the `Switchboard` itself dies, a new one may be started that recovers the state at the previous failure. In such a case, all the decoders and `DecoderServers` will wait until the new `Switchboard` is ready and then re-register and resume processing sentences.

4.3.1 Language package

The most significant data encapsulation of the design is that of the language package. The language package, which will typically be an actual Java package,⁴ is a collection of Java classes that are implementations of several interfaces which provide the specification of data and methods specific to a particular language and Treebank annotation style.⁵ There are four such required classes: `Treebank`, `Training`, `HeadFinder` and `WordFeatures`. The `Treebank` class provides all data and methods specific to a particular Treebank, such as predicates for preterminals and complex nonterminal labels. The `Training` class provides methods and data specific to preprocessing training trees. The `HeadFinder` class' primary function is to read a data (text) file specifying head rules specific to a particular language's Treebank and provide a head-finding method. Finally, the `WordFeatures` class provides a mapping of lexical items of a particular language to orthographic/morphological word-feature-vectors, to aid part-of-speech tagging.⁶ As an additional language-independent design criterion, all relevant input and output files are written to and read from with a user-settable character encoding.⁷ Our goal was, for treebanks that are not too radically different from the Penn Treebank (such as the Chinese Treebank (Xia et al., 2000)), to be able to create a new language package in about 1–2 weeks. As it happens, for Chinese, the time to create a language package was one and a half days. For other languages, the time has generally been in the 1–2 week range, with most of the time spent dealing with data representation issues, rather than deeper linguistic issues.

⁴A Java *package*, much like its Lisp progenitor, provides a separate name space for a collection of classes and their data members, as well as providing yet another level of access control for data and methods (i.e., data and methods can be made accessible only within their enclosing package).

⁵An *interface* is a Java construct (borrowed from the Smalltalk programming language) that specifies a contract for implementations to follow, in the form of a set of method (function) signatures, augmented as needed by API documentation.

⁶Unlike (Collins, 1997; Collins, 2000), part-of-speech tagging can be fully integrated into the models supported by our architecture. Along these lines, word features have been shown to reduce part of speech ambiguity for unknown words (Weischedel et al., 1993) and may be employed by models in our architecture.

⁷Java stores all characters internally as unicode characters, and most run-time environments support a very wide variety of character encodings, supporting an even wider variety of the world's languages, making Java an ideal language with which to implement internationalizable software.

Relating this encapsulation to Chapter 3, the generative model itself is completely language-independent, and, except for the comma-pruning constraint, so is the decoding algorithm. However, as we have seen, much of the important preprocessing of trees—producing features that are explicitly modeled—is language-dependent. In order to preserve the language independence of the core of the engine (the model and the decoder), the following is a partial list of language- and/or Treebank-dependent features that require encapsulation via the language package, along with back-pointers to the relevant sections of Chapter 3:

- head-finding rules (§3.3.12)
- argument-finding rules (§3.3.9)
- node/requirement types that can appear in subcats (§3.5.2)
- commas and other types of punctuation (§§3.3.8 and 3.6.2)
- nodes that represent sentences (§§3.3.6 and 3.3.11)
- predicates for base NPs (§3.3.3)
- words/nodes to delete (§3.3.2)

Chapter 5 presents the results of extending the engine to new languages, and in Chapter 7 we will explore detailed differences between the Chinese and English models.

4.3.2 Probability structure objects

As we saw in §3.5.7, each parameter class has an associated suite of mapping functions ϕ_i , $0 \leq i < n$, where n is the number of back-off levels. Both the BBN SIFT parser and Collins parser essentially hard-coded these mapping functions, with the BBN parser going so far as to have a separate data type for each back-off level of each parameter class.

In our design, every type of output element of the models supported by our engine—including nonterminals, preterminals (parts of speech), words, word-features, gaps and subcat frames—has

an associated `ProbabilityStructure` object, that specifies how to form the data objects representing the future and history at all possible back-off levels for that output element. Objects containing maximal context—called `TrainerEvent` objects—are passed in to a single method which then delivers the appropriate history or future for a specified back-off level, as per the pseudocode example in §4.2.1:

```
history = probabilityStructure.get(backOffLevel, trainerEvent);
```

A layer of abstraction also exists for these probabilistic events: objects that represent events must conform to an `Event` interface. This abstraction allows for maximum flexibility in the underlying types and representation of event elements, which are typically the output elements of the parser, but can also include other features derived from those output elements. In this way, the user can not only easily test new probability structures in our architecture, but also new events with new *types* of event elements. Having this layer of abstraction made quick work of the dependency-removal experiments of §3.7.

4.3.3 Flexible subcat frames

As we saw in §3.5.2, Collins’ model uses a kind of subcategorization frame (subcat) that is a multiset of requirements on a particular side of a head child. This is a dynamically-updated subcat, meaning that when a requirement is satisfied, it is removed from the multiset. For example, if a left subcat consists solely of `{NP-A}`, and an `NP-A` node is generated on the left-hand side of a head child, then subsequently-generated left modifiers will condition on an empty subcat, `{}`. In this way, the dynamic subcat acts as kind of a history-based model within the history-based model.

In our engine, subcat frames are implementations of a very simple yet powerful `Subcat` interface. During training, all (unlexicalized) modifiers of a head child in every production are passed to the subcat’s `add` method; the subcat implementation may choose to ignore any or all of these nonterminals. It may collect certain of them—such as those augmented with `-A`—into a multiset, as Collins’ model does, it may collect them into an ordered list, as is effectively done by Chiang’s stochastic tree-insertion grammar model (Chiang, 2000), or it may simply collect the *number* of such argument children, constraining only the number of arguments on a particular side of a

head child. During decoding, all modifiers generated by the model are passed to the subcat implementation's `remove` method, which will equivalently act upon or ignore each nonterminal. In our `SubcatBag` class that implements Collins-style subcats, a counter associated with each of five types of arguments (NP-A, S-A, SBAR-A, VP-A and miscellaneous) is decremented as they are generated and passed to the `remove` method during decoding.

4.3.4 Flexible constraint-satisfaction system for constrained parsing

The only hard constraints in a probabilistic parsing model are events that never occurred in training and thus have a probability estimate of 0. However, some or all of these events can be given *some* probability mass, thereby turning them back into soft constraints as with every other event of the model. In spite of the great success of statistical models to decide among competing analyses with soft constraints, there are times when it is useful or necessary to specify hard constraints, such that the parser only delivers parses that are consistent with those constraints. The parsing engine described here provides a flexible mechanism for the specification of such constraints.

Suppose, for example, that there were phrases, collocations or named entities in data that were already marked—either by humans or automatically—with a high degree of accuracy. Such phrases or collocations may be fed to the engine as constraints, so that it would only produce parses that contained those phrases. Alternatively, suppose there were a “clause detector” that demarcated clauses within sentences. In such a case, it might be desirable to have parses with brackets that never crossed these clause boundaries; the engine is also capable of parsing with this type of constraint. In fact, several constraint systems are provided with the engine, and the type of constraint used when parsing, if any, is selectable via a run-time setting. Developing new constraint types for the engine is also straightforward.

Another use for constrained-parsing is for bootstrapping the creation of Treebanks. One strategy would be to have annotators fully annotate a small amount of “seed” training data in a new language or domain, on which the parsing engine would be initially trained. Then, the annotators could focus only on marking, say, noun phrases in the full set of data. The parsing engine could “fill in” the rest of the syntactic structure by producing parses consistent with the already-marked

phrases. Then, the annotators could correct all phrases of another type (say, verb phrases), the parser could be re-trained on this corrected data, and the cycle could continue until all types of phrases were correctly marked.

Finally, constraints allow the researcher to treat some part of the joint probability model as “hidden data”, using the Inside-Outside algorithm as described in(Lari and Young, 1990; Pereira and Schabes, 1992). This scheme maximizes the likelihood of the model subject to a given set of constraints. The parsing engine not only provides the necessary constrained-parsing mechanism, but crucially also has an implementation of this Expectation-Maximization algorithm.

There is also a scientific/engineering use for constrained parsing that has repeatedly shown itself to be invaluable. There are times when the decoder cannot deliver a parse for a particular sentence. Sometimes this has to do with pruning, but often it is because there was an unexpected zero-probability estimate for an event without which a tree could not be constructed (the hard constraints discussed above). When one has access to the gold-standard tree, one can have the parser constrain-parse it and output which constraints were not satisfied, and thus pinpoint precisely the substructures(s) that could not be derived. This is not just a debugging aid (the engineering use), but is also a grammar-modeling aid, as it may point out previously unknown limitations in a particular grammar or model of that grammar.

4.4 Efficiency

4.4.1 Probability-level parallelism

The simplest form of parallelization is to increase a parser’s throughput by splitting up a test file into multiple segments and have multiple executables run on these segments, each on a separate host. Our sentence server provided a finer granularity of parallelism, but still required a separate parser on each host. Schemes such as these are a monumental waste of memory resources: each parser contains the same massive tables containing counts of events observed in the training data. Our current design exhibits a much finer degree of parallelism by introducing a *probability server*. The idea is that a `DecoderServer` object sits on a multiprocessor with a large amount of RAM,

serving up smoothed top-level probability estimates to multiple, small chart parsing clients. The architecture also supports load-balancing if multiple, large multiprocessing hosts are available. Java provides an elegant solution to such distributed computing via a built-in technology called *remote method invocation*, or RMI.

4.4.2 Optimizations

We did not want all the flexibility of the engine to compromise the efficiency of our system; thus, we spent significant effort optimizing the decoder.⁸

1. optional pre-computation of all log-probability estimates and log-lambdas, necessitating only floating-point add operations for computing probability estimates during decoding
2. creation and use of hash maps optimized for mapping arbitrary Java objects to primitive types, achieving spatial locality in probability lookups and eliminating the need for primitive wrapper objects
3. short-circuiting all decoding operations when it is evident that the resulting new chart item will have a zero probability
4. elimination of almost all new-object creation during decoding, including the use of an object pool for efficient recycling of chart item objects
5. smaller optimizations based on profiling

Optimization technique 3 was of crucial importance, achieving the single greatest speed improvement of them all. In particular, before the computation of the modifier-generation probability, the decoder first checks to see if the last level of back-off for the modifier-generation model (the history) ever occurred with the particular modifier being generated (the future) in training. If not, then it will be impossible for the model to deliver anything but a zero probability. Put another way, if the coarsest context never occurred with the future being generated, then certainly more specific contexts could never have occurred, and thus the model would deliver a zero probability for the

⁸Our thanks to Scott Miller for his invaluable advice for parser optimization.

entire smoothed estimate. This optimization alone resulted in a 5x speedup, and so it was also implemented for the subcat-generation model, resulting in an even greater speedup.

A note on RMI

Remote method invocations via RMI can have a large amount of overhead, even between two hosts on the same local-area network; the overhead can occasionally approach a few milliseconds. Such delays would present a significant bottleneck to parsing efficiency, given the enormous number of probability requests the client makes for just a single sentence (100,000 probability requests would not be uncommon). However, experiments have shown that even a modestly-sized cache on the client host results in no more than a 2x slowdown over having the DecoderServer and client on the same host, using two Linux machines on the same 100baseT LAN.

4.5 Summary

In the design presented here, we have shown how using various well-known engineering practices and technologies of computer science, we can address several limitations of other statistical parsers. Indeed, almost all of the techniques and technologies we used here can be applied to NLP software in general, particularly in light of the growing need to develop efficient, internationalizable software. By abstracting away from the idiosyncrasies of training data and from a particular probability structure for parser output elements, we provide a means to test various probability structures, using features and back-off schemes that are the result either of pure theorizing or empirical study or (presumably) both, and we allow this experimentation while maintaining a high degree of computational efficiency. As has been shown in (Collins, 2000), choosing linguistically-motivated features and independence assumptions can often result in a well-estimated, high-performance model. It is thus our hope that continued application of sound engineering practices will not only lead to more efficient, scalable and extensible NLP software, but will also benefit the exploration and testing of linguistic theories and their associated models.

Chapter 5

Parsing in multiple languages

In (Bikel and Chiang, 2000), we describe porting two different generative models to Chinese, coinciding with the release of the Chinese Treebank. We will review the details of this porting process, an experience which, in part, inspired the extensible design of the current parsing engine described in Chapter 4. Next, we will present our most recent Chinese parsing results using our Chinese language package and (most parts of) the core Collins model. In §5.4, we discuss our experience with the evolving Arabic Treebank project, and finally we'll look at the issues encountered when extending the engine to Classical Portuguese for an historical linguistics project.

5.1 Porting English parsers to Chinese

5.1.1 Models and modifications

The two models we ported were BBN's SIFT model (Miller et al., 1998) and the stochastic TIG model of Chiang (2000). Both models inherit a great deal from our baseline model, Collins' Model 2.

The primary language-dependent component that had to be changed in both models was the head table, used to determine heads when training. We modified the head rules described in (Xia, 1999) for the Xinhua corpus and substituted these new rules into both models.

The (Chiang, 2000) model had the following additional modifications.

- The new corpus had to be prepared for use with the trainer and parser. Aside from technicalities, this involved retraining the part-of-speech tagger described in (Ratnaparkhi, 1997), which was used for tagging unknown words. We also lowered the unknown word threshold from 4 to 2 because the Xinhua corpus was smaller than the WSJ corpus.
- In addition to the change to the head-finding rules, we also changed the rules for classifying modifiers as arguments or adjuncts. In both cases the new rules were adapted from (Xia, 1999).
- For the tests done in this section, a beam width of 10^4 was used.

The BBN model had the following additional modifications:

- As with the (Chiang, 2000) model, we similarly lowered the unknown word threshold of the BBN model from its default 5 to 2.
- The original model associated a word-feature vector with every word, largely to alleviate part-of-speech ambiguity for unknown words. This word-feature vector was eliminated, causing parts of speech for unknown words to be predicted solely on the head relations in the model.
- The default beam size in the probabilistic CKY parsing algorithm was widened. The default beam pruned away chart entries whose scores were not within a factor of e^5 of the top-ranked subtree; this tight limit was changed to e^9 . Also, the default decoder pruned away all but the top 25-ranked chart entries in each cell; this limit was expanded to 50.

In addition, there was the technical challenge of “ferreting out” all English-specific assumptions in the source code of both parsers.

5.1.2 Initial results

The initial release of the Chinese Treebank consisted of 4185 sentences of Xinhua newswire text. We blindly separated this into training, devtest and test sets, with a roughly 80/10/10 split, putting

Model, test set	≤ 40 words					
	LR	LP	$\overline{\text{CB}}$	0CB	$\leq 2\text{CB}$	F
BBN-all [†] , WSJ-all	84.7	86.5	1.12	60.6	83.2	85.6
BBN-small [†] , WSJ-small*	79.0	80.7	1.66	47.0	74.6	79.8
BBN, Xinhua[‡]	69.0	74.8	2.05	45.0	68.5	71.8
Chiang-all, WSJ-all	86.9	86.6	1.09	63.2	84.3	86.7
Chiang-small, WSJ-small	78.9	79.6	1.75	44.8	72.4	79.2
Chiang, Xinhua	76.2	77.2	2.09	49.8	71.8	76.7
	≤ 100 words					
	LR	LP	$\overline{\text{CB}}$	0CB	$\leq 2\text{CB}$	F
BBN-all [†] , WSJ-all	83.9	85.7	1.31	57.8	80.8	84.8
BBN-small [†] , WSJ-small*	78.4	80.0	1.92	44.3	71.3	79.2
BBN, Xinhua[‡]	67.5	73.5	2.87	39.9	61.8	70.4
Chiang-all, WSJ-all	86.2	85.8	1.29	60.4	81.8	86.0
Chiang-small, WSJ-small	77.1	78.8	2.00	43.25	70.5	77.9
Chiang, Xinhua	72.2	73.9	3.08	43.9	64.7	73.0

Table 5.1: Results for both parsing models on all test sets. Preterminal nodes were not counted as brackets for the purposes of computing labeled precision and recall. Key: LR = labeled recall, LP = labeled precision, $\overline{\text{CB}}$ = avg. crossing brackets, 0CB = zero crossing brackets, $\leq 2\text{CB}$ = ≤ 2 crossing brackets, F = the evenly-weighted harmonic mean of LR and LP. All results are percentages, except for those in the $\overline{\text{CB}}$ column. [†]Used larger beam settings and lower unknown word threshold than the defaults. *3 of the 400 sentences were not parsed due to timeouts and/or pruning problems. [‡]3 of the 348 sentences did not get parsed due to pruning problems, and 2 other sentences had length mismatches (scoring program errors).

files 001–270 (3484 sentences, 84,873 words) into the training set, 301–325 (353 sentences, 6776 words) into the development test set and reserving 271–300 (348 sentences, 7980 words) for testing. See Table 5.1 for results.

In order to put the new Chinese Treebank results into context with the unmodified (English) parsing models, we present results on two test sets from the Wall Street Journal: WSJ-all, which is the complete Section 23 (the *de facto* standard test set for English parsing), and WSJ-small, which is the first 400 sentences of Section 23 and which is roughly comparable in size to the Chinese test set. Furthermore, when testing on WSJ-small, we trained on a subset of our English training data roughly equivalent in size to our Chinese training set (Sections 02 and 03 of the Penn Treebank);

parent pattern	child pattern list
VP	IP, CP, VP
CP	IP, CP
PP	NP, LCP, QP, DP, IP, CP, UCP
DNP	DP, QP, NP, LCP, PP, ADJP, UCP
DVP	DP, NP, QP, VP, ADVP, UCP
LCP	NP, QP, DP, LCP, PP, IP, UCP
*	*-SBJ, *-OBJ, *-IO, *-PRD

Table 5.2: Heuristics for marking arguments in the Chinese Treebank.

we have indicated models trained on all English training with “-all”, and models trained with the reduced English training set with “-small”. Therefore, by comparing the WSJ-small results with the Chinese results, one can reasonably gauge the performance gap between English parsing on the Penn Treebank and Chinese parsing on the Chinese Treebank.

The reader will note that the modified BBN model does significantly more poorly than (Chiang, 2000) on Chinese. We suspect part of the difference may be due to the fact that the BBN model uses language-specific rules to guess part of speech tags for unknown words.

5.2 Current Chinese results

To extend our parsing engine to Chinese, we created a language package and used the core of our “Collins emulation” model. In addition to the tweaked head-finding rules of Xia (1999) (see Appendix A for a complete listing), we needed to create argument-finding heuristics, so as to make use of the subcat features of the model. We used essentially the same heuristics employed by Chiang (partly for comparison testing), and these are shown in Table 5.2. If a node’s parent matches any of the parent patterns, then the node is matched against each of the corresponding child pattern lists for which its parent matched. If the node matches one or more of the child patterns, then it is marked as an argument unless it bears the ADV function tag. All nonterminals match the * wildcard, and all nonterminals with the SBJ augmentation match *-SBJ, and similarly for other patterns of the form “*-AUG”, where AUG is some node augmentation. Also, patterns without augmentations,

such as NP, subsume nodes that have augmentations, such as NP-SBJ.

As an example, if the node under scrutiny is NP-SBJ and its parent is PP, then the NP-SBJ's parent would match both the PP and the * parent patterns. Next, the NP-SBJ would be checked against the argument child patterns for PP and *. Sure enough, the PP's child pattern list contains the pattern NP, which matches NP-SBJ via subsumption, *and* the child pattern list for * contains *-SBJ, which also matches NP-SBJ. So, since there is at least one matching pattern, and since the NP-SBJ does not bear the -ADV augmentation, it is considered an argument by these heuristics.

There were other, treebank-specific pieces of information that were required to create the Chinese language package (and that are required to create any language package). For example, in order to implement the comma pruning rule (the use of which is run-time switchable), it was necessary to determine the Unicode representations for Chinese commas. In order to prune away quotation marks, which, like the English Treebank, were given low priority with respect to placement inside phrases, it was necessary to determine the wide range of characters that serve this function (there are, in fact, no fewer than *seven* varieties of quotation marks in the Chinese Treebank).

Finally, it was necessary to determine appropriate or optimal values for the large number of run-time settings provided by our parsing engine. These included not only the comma-pruning setting mentioned above, but also the unknown word threshold, the beam width and others.¹ In contrast to the initial experiments, we set the beam width to 10^5 .

There are currently three versions of the Chinese Treebank available, 1.0, 3.0 and 4.0. Each new version is a superset of the data of the previous version. The sizes of our training and test corpora for these versions are listed in Table 5.3. While versions 1.0 and 3.0 consist solely of Xinhua newswire articles, version 4.0 adds articles from Sinorama magazine. The results of our engine on the Chinese Treebank, versions 1.0, 3.0 and 4.0 are listed in Table 5.4.²

¹The file `chinese.properties` that is part of the installation of the parsing engine software contains the complete list.

²Three formatting errors were corrected with the CTB3.0 test data: in sentences 27 and 47, two null element words were missing the -NONE- preterminal in their trees, and in sentence 547 a closing parenthesis was assigned the "part-of-speech tag" of NP, which is not among the set of part-of-speech tags that serve as preterminals. While it is generally bad practice to correct held-out, blind test data, I felt reasonably justified in correcting formatting errors that could be identified by any automatic tree format verifier (in fact, that is how I caught these errors).

Treebank	training			test	
	sentences	tokens	types	sentences	tokens
1.0 [†]	3485	84,937	9588	348	7980
3.0	9605	230,559	19,114	775	20,201
4.0	12,334	331,110	28,133	1378	35,743

Table 5.3: Sizes of training and test corpora for the three available versions of the Chinese Treebank. Tokens and types denote numbers of words in the data. [†]Statistics are slightly different from those reported in §5.1.2, since one of the sentences was removed from the data set for some reason lost to history.

Treebank	≤ 40 words					
	LR	LP	$\overline{\text{CB}}$	0 CB	$\leq 2\text{CB}$	F
1.0	78.0	81.2	1.55	53.8	76.1	79.6
3.0	79.6	82.9	1.36	50.8	80.1	81.2
4.0	76.9	81.1	1.77	46.8	72.6	78.9
	all sentences					
	LR	LP	$\overline{\text{CB}}$	0 CB	$\leq 2\text{CB}$	F
1.0	74.4	78.5	2.53	47.4	69.3	76.4
3.0 [†]	77.5	80.6	2.10	42.5	70.2	79.0
4.0	74.7	79.0	2.78	39.6	63.1	76.8

Table 5.4: Current results on the Chinese Treebank, versions 1.0, 3.0 and 4.0. [†]CTB3.0 results are only for sentences of length ≤ 100 words.

5.3 Discussion of Chinese results

After the initial results on the first version of the Chinese Treebank, we found that a lot of errors appeared to center around paratactic NP constructions, and these constructions also appeared to be the source of a great deal of parse ambiguity, leading to a dramatic decrease in parsing efficiency. Another source of ambiguity and error that we observed appeared to be the abundance of coordinated phrases. Errors involving these constructions and more were also observed and analyzed in detail by Levy and Manning (2003), who devised heuristic node transformations (as per the work of Johnson (1998)) to alleviate the ambiguity, *e.g.*, with NPs modifying other NPs. Levy et al. were employing a factored model (Klein and Manning, 2002b), which combines an unlexicalized PCFG model with a dependency model involving no constituent labels. They found a significant improvement in parse accuracy with their grammar transformations, which dealt not only with the NP-NP modification problem, but also with the problem of the parser producing multilevel VPs when the corpus called for flatter VP structures, as well as four other types of errors. Most of the performance gains due to the grammar transformations, as might be expected, were in recall and not precision (in fact, precision actually *decreased* by a very small amount). However, we note that even with these performance gains, their parser achieves a lower F-score than ours on the same data.

While CTB4.0 is still not in its final form, it is still disappointing that the parsing results are inferior to those for CTB3.0, in spite of CTB4.0 being a 60% larger data set.³ We speculate that the multiple sources for the CTB4.0 data is adding undesirable noise (an effect similar to that noticed by Gildea (2001) for parsing multiple sources of English). It is also quite possible that further tree correction by the annotators will improve parsing scores. On the other hand, it is encouraging that both the CTB1.0 and CTB3.0 results are clearly superior to all the results reported in §5.1.2.

Part-of-speech tagging is an important issue, since the engine uses the supplied part-of-speech tags for unseen words. As with the initial Chinese experiments, we used `mxpost` (Ratnaparkhi, 1996) having been re-trained each time for the training set for each version of the CTB. While this re-training was simple, `mxpost` contains English-specific morphological information. It therefore

³All observed differences in precision and recall on the three data sets are statistically significant with $p < 0.01$ in all cases, as per a randomized stratified shuffling test (Cohen, 1995).

Test set	≤ 40 words					F
	LR	LP	$\overline{\text{CB}}$	0 CB	$\leq 2\text{CB}$	
CTB3.0	79.6	82.9	1.36	50.8	80.1	81.2
CTB3.0-gold-tags	80.9	84.5	1.28	53.0	81.4	82.7
all sentences						
CTB3.0	77.5	80.6	2.10	42.5	70.2	79.0
CTB3.0-gold-tags	79.2	82.3	2.01	44.2	71.0	80.7

Table 5.5: Comparison of results on parsing the CTB3.0 test set using output of MXPOST versus gold-standard part-of-speech tags for unseen words.

has a model that is (presumably) sub-optimal for Chinese, especially for unknown words, where language-specific word features are the only lexical cues available to augment that tag n -grams in the model. To gauge the effect of unseen-word tagging accuracy on parsing performance, we did an experiment with the CTB3.0 training/test split, using gold-standard parts-of-speech when providing test sentences to our parser. As always, the parser does its own part-of-speech tagging for words that it has seen in training. The results are shown in Table 5.5. These results are statistically significant in both precision and recall compared to the baseline CTB3.0 results reported in Table 5.4 ($p < 0.001$ for recall, $p < 0.004$ for precision, using a randomized stratified shuffling test (Cohen, 1995), with even lower p -values when comparing scores for sentences of length ≤ 40 words).

All of the parsing work in Chinese reported here has ignored a crucial issue in that language, segmentation. We have, all along, been using the gold-standard, hand-segmented data from the CTB. A real-world Chinese parser would need to use a segmenter as a preprocessor, or combine segmentation, tagging and parsing into a single model. Bert Xue of the CTB project has been working on such a system.

5.4 Other languages

There are fewer experiments possible with the other phrase-structure treebanks that are becoming available, because they are still small and in a great state of flux. Nevertheless, we report

on our experiences with the Arabic Treebank and on an historical linguistics project on Classical Portuguese. This latter project was joint work with Anthony Kroch, Chair of the Linguistics Department at the University of Pennsylvania, and with linguists from the University of Campinas in São Paulo, Brazil.

5.4.1 The Arabic Treebank

Arabic has a much richer morphology than English, including not only morphology to indicate syntactic function, but also cliticization of pronouns and other forms. Thus, deciding on a part-of-speech tag set was one of the first important decisions of the project. In the end, the tag for each word was manually chosen from a set of candidates from a morphological analyzer built by Tim Buckwalter of the Linguistic Data Consortium. In the format of the analyzer, a word form consisting of multiple, “constituent” word forms would be analyzed with the individual constituents’ parts of speech concatenated with a plus sign. For example, since determiners can be attached to their nouns, a common part-of-speech tag is DET+NOUN_PROP, indicating a proper noun with a preceding determiner. While using the output of the Buckwalter analyzer had the practical consequence of getting the project up and running quickly, it had the undesirable consequence—from a modeling point of view—of creating a very large tagset that was not even static, for if a new word form appeared that contained several cliticized forms that had not been seen together before, a new tag would be “created”. This pointed out a limitation inherent to many Anglo-centric NLP models: the expectation that $\hat{p}(w|t) > 0$ for all known words with their correct tags.

The goal of parsing in this project was not for some further downstream use such as semantic analysis or machine translation, but to help bootstrap the creation of more parsed data. In order to quickly get the core parsing model running, we created a language package that provided a mechanism to map the productive tagset to a small, static set, on the order of the size of the English tagset. After experimenting with several different mapping schemes, Ann Bies, a linguist with the original Penn Treebank project (Bies, 1995) as well as a member of the Arabic Treebank project staff, provided a “gold-standard” mapping that is now included with the distribution of the data.

We separated the original data, which came from the news source AFP, into training and test

≤ 40 words					
LR	LP	\overline{CB}	0 CB	$\leq 2CB$	F
75.4	76.0	3.00	28.1	57.6	75.7
all sentences					
72.5	73.4	4.16	23.7	50.3	72.9

Table 5.6: Parsing performance on an early version of the Arabic Treebank, using a mapped version of the gold-standard part-of-speech tags.

(due to the fact that this project was a moving target, we did not bother to create a development test set). Articles 0000–0150 (3623 sentences, 149,379 word tokens) were set aside as training, and articles 0150–0236 (388 sentences, 11,176 word tokens) were used as held-out test data. We only used the unvoveled data, because that would ultimately be necessary for any real-world parser. However, because the purpose was for bootstrapping treebank annotation, we used the gold-standard, Bies-mapped part-of-speech tags. The scores with this set-up are shown in Table 5.6.

5.4.2 Classical Portuguese

The historical linguistics project undertaken by the Linguistics Department at the University of Campinas was intended to produce parsed corpora of Portuguese in various stages of that language’s historical development. The goal was to uncover reasons for language change by studying corpus statistics of a changing phenomenon diachronically, following the approach that had proven successful for English (Kroch, 2000). For bootstrapping creation of the Classical Portuguese treebank, the Brazilian annotators created a treebank from about 43k words of text, to be used as training for our parser, which would then automatically mark up new data that had been part-of-speech tagged, so that the annotators would only need to correct the automatic trees, instead of starting from scratch.

Portuguese, like Arabic, has a rich morphology compared to English. In spite of this, the part-of-speech tag set was kept to a fairly small number, comparable to the size of the English tag set. Nevertheless, an issue arose related to the $\hat{p}(w|t)$ issue for Arabic, pointing out another tacit assumption of many Anglo-centric models of syntax. Just as with English (and, presumably,

all other languages), certain of the part-of-speech classes of Portuguese are closed, meaning they consist of only a fixed, finite number of words (the classes can change, but only very slowly over time). One such closed class is for pronouns. As it happens, none of the first batch of data on which we trained our parser contained an example of the 2nd person plural pronoun *vos*. When the second batch of data arrived to be parsed, we trained our engine on the first batch and parsed the second, to discover that an alarming number of sentences were receiving no parse at all. This turned out to be due to the fact that every instance of *vos* was being mapped to the special +UNKNOWN+ token, since it was an unseen word, but when $\hat{p}(+UNKNOWN+|CL)$ was computed (CL is the correct tag for *vos*), an estimate of 0 was produced, because +UNKNOWN+ was never seen in training occurring with a closed-class part-of-speech tag, as one would expect.

Kroch observed that syntactically, the actual word probably did not matter for such closed-class items, meaning that *any* estimate of $p(w|CL)$ would do. Our solution, therefore, was to create yet another run-time option for the parsing engine whereby for any word w and closed-class tag t_c , if $\hat{p}(w|t_c) = 0$, it would select at random another word from the same closed class and use that estimate instead. This is a robustness feature that should typically only be useful in the face of very small data sets (such as we had with Classical Portuguese), but might also be useful in other situations.

While the parser was initially a success, the project has experienced multiple periods of inactivity, and, to our knowledge, remains in a dormant state.⁴

5.4.3 Current projects

A Master's student at the University of Pennsylvania, Jinho Choi, wrote a language package for the parsing engine to use the evolving Korean Treebank, with some encouraging initial results. Abhishek Arun, a student at the University of Edinburgh, has written a language package for the newly-released French Treebank, consisting of articles from the newspaper *Le Monde*. Prof. Daniel Gildea of the University of Rochester has been exploring the use of the parsing engine for syntax-based Chinese/English machine translation, looking at methods for doing tree-based alignment.

⁴Ideally you, the reader, are not yet in a dormant state.

Chapter 6

Independence assumptions, back-off and meta-parameters

6.1 Further discussion of the baseline model

Chapter 3 accounts for the noticeable effects of all the unpublished details of Collins’ model. But what of the details that *were* published? In Chapter 8 of his thesis, Collins gives an account on the motivation of various features of his model, including the distance metric, the model’s use of sub-cats (and their interaction with the distance metric) and structural versus semantic preferences. In the discussion of this last issue, Collins points to the fact that structural preferences—which, in his model, are modeled primarily by the P_L and P_R parameters—often provide the right information for disambiguating competing analyses, but that these structural preferences may be “overridden” by semantic preferences. Bilexical statistics (Eisner, 1996), as represented by the maximal context of the P_{L_w} and P_{R_w} parameters, serve as a proxy for such semantic preferences, where the actual modifier word (as opposed to, say, merely its part of speech) serves to indicate the particular semantics of its head. Indeed, such bilexical statistics were widely assumed for some time to be a source of great discriminative power for several different parsing models, including that of Collins.

However, Gildea (2001) had re-implemented Collins’ Model 1 (essentially Model 2 but without

Back-off level	Number of accesses	Percentage
0	3,257,309	1.49
1	24,294,084	11.0
2	191,527,387	87.4
Total	219,078,780	100.0

Table 6.1: Number of times our parsing engine was able to deliver a probability for the various levels of back-off of the mod-word generation model, P_{M_w} , when testing on Section 00, having trained on Sections 02–21. In other words, this table reports how often a context in the back-off chain of P_{M_w} that was needed during decoding was observed in training.

subcats) and had altered the P_{L_w} and P_{R_w} parameters so that they no longer had the top level of context that included the head word (he removed “Back-off level 0”, as depicted in Table 3.1). In other words, Gildea removed all bilexical statistics from the overall model. Surprisingly, this resulted in only a 0.45% absolute reduction in F-measure (3.3% relative increase in error). Unfortunately, this result was not entirely conclusive, in that Gildea was only able to do a partial re-implementation of Collins’ baseline model, the performance of which was not quite as good as that of Collins’ parser.¹ Training on Sections 02–21, we have duplicated Gildea’s bigram-removal experiment, except that our chosen test set is Section 00 instead of Section 23 and our chosen model is the more widely-used Model 2. Using the mode that most closely emulates Collins’ Model 2, with bigrams, our engine gets a recall of 89.89% and a precision of 90.14% on sentences of length ≤ 40 words (see Table 6.3, Model $\mathcal{M}_{tw,tw}$). Without bigrams, performance drops only to R89.49%, P89.95%—an exceedingly small drop in performance (see Table 6.3, Model $\mathcal{M}_{tw,t}$). In an additional experiment, we examined the number of times that the parser was able to deliver a requested probability for the modifier-word generation model using the increasingly less-specific contexts of the three back-off levels, while decoding Section 00. The results are in Table 6.1. Back-off level 0 indicates the use of the full history context, which contains the head child’s head word. Note that probabilities making use of this full context, that is making use of bilexical dependencies, are used only 1.49% of the time. Combined with the results from the previous experiment, this suggests that such statistics

¹The re-implementation was necessarily only partial, as Gildea did not have access to all the unpublished details of Collins’ models that are presented in Chapter 3.

are far less significant than once thought to the overall discriminative power of Collins’ models, confirming Gildea’s result for Model 2.²

6.1.1 Choice of heads

If not bilexical statistics, then surely, one might think, head-choice is critical to the performance of a head-driven, lexicalized statistical parsing model. Partly to this end, in §6.3 we will explore methods for recovering latent information in treebanks, where one of the ideas is to use the Inside–Outside algorithm to reestimate the parameters of a model defined over an *augmented tree space*, where the observed data are the gold-standard labeled bracketings found in the Treebank, and the hidden data are the head lexicalizations, one of the most notable tree augmentations performed by modern statistical parsers. The promise of using EM is that very little effort would need to be spent on developing head rules, since EM could take an initial model that used simple heuristics and optimize it appropriately to maximize the likelihood of the unlexicalized (observed) training trees. To test this, we performed experiments with an initial model trained using an extremely simplified head-rule set, where all rules were of the form “if the parent is X, then choose the left/rightmost child”. A surprising side result was that, even with this simplified set of head rules, overall parsing performance still remained quite high. Using our simplified head-rule set for English, our engine in its “Model 2–emulation mode” achieved a recall of 88.55% and a precision of 88.80%, for sentences of length ≤ 40 words in Section 00 (see Table 6.2). So, contrary to our expectations, the lack of careful head choice is not crippling in allowing the parser to disambiguate competing derivations. Furthermore, given the simplicity with which head nodes are marked—and hence the simplicity with which head words are chosen—this result calls into question the degree to which semantic preferences, as represented by conditioning on a head word, override structural preferences.

²On a separate note, it may come as a surprise that the decoder needs to access more than 219 million probabilities during the course of parsing the 1917 sentences of Section 00. Among other things, this certainly points to the utility of caching probabilities (the 219 million are tokens, not types).

Model	LR	LP	CBs	0 CBs	≤ 2 CBs	F
Collins' Model 2	89.75	90.19	0.77	69.10	88.31	89.97
Baseline (Model 2 emulation)	89.89	90.14	0.78	68.82	89.21	90.01
Simplified head rules	88.55	88.80	0.86	67.25	87.42	88.67

Table 6.2: Results on Section 00 with simplified head rules. The baseline model is our engine in its closest possible emulation of Collins' Model 2.

Parameter class		P_M		P_{M_w}		Score					
conditioning on		t_h	w_h	t_h	w_h	LR	LP	CBs	0 CBs	≤ 2 CBs	F
Model	$\mathcal{M}_{tw,tw}$	✓	✓	✓	✓	89.89	90.14	0.78	68.82	89.21	90.01
	$\mathcal{M}_{tw,t}$	✓	✓	✓		89.49	89.95	0.80	67.98	88.82	89.72
	$\mathcal{M}_{t,t}$	✓		✓		88.20	88.89	0.91	65.00	87.13	88.54
	$\mathcal{M}_{tw,\emptyset}$	✓	✓			89.24	89.86	0.81	66.80	88.76	89.55
	$\mathcal{M}_{t,\emptyset}$	✓				88.01	88.96	0.91	63.93	86.91	88.48
	$\mathcal{M}_{\emptyset,\emptyset}$					87.01	88.75	0.96	61.08	86.00	87.87

Table 6.3: Parsing performance with various models on Section 00 of the Penn Treebank. P_M is the parameter class for generating partially lexicalized modifying nonterminals (a nonterminal label and part of speech). P_{M_w} is the parameter class that generates the head word of a modifying nonterminal. Together, P_M and P_{M_w} generate a fully-lexicalized modifying nonterminal. The check marks indicate the inclusion of the head word w_h and its part of speech t_h of the lexicalized head nonterminal $H(t_h, w_h)$ in the conditioning contexts of P_M and P_{M_w} .

6.1.2 Lexical dependencies matter

Given that bilexical dependencies are almost never used and have a surprisingly small effect on overall parsing performance, and given that the choice of head is not terribly critical either, one might wonder what power, if any, head lexicalization is providing. The answer is that, even when one removes bilexical dependencies from the model, there are still plenty of *lexico-structural* dependencies, *i.e.*, structures—subcats and partially-lexicalized modifiers—being generated conditioning on head words, and head words being generated conditioning on structures.

To test the effect of such lexico-structural dependencies in our lexicalized PCFG-style formalism, we experimented with the removal of the head tag t_h and/or the head word w_h from the conditioning contexts of the P_{M_w} and P_M parameters. The results are listed in Table 6.3. Model $\mathcal{M}_{tw,tw}$ shows our baseline, and Model $\mathcal{M}_{\emptyset,\emptyset}$ shows the effect of removing *all* dependence on the head word and its part of speech, with the other models illustrating varying degrees of removing elements from the two parameter classes' conditioning contexts. Notably, including/removing the

head word w_h from the P_M contexts appears to have a significant effect on overall performance, as shown by moving from Model $\mathcal{M}_{tw,t}$ to Model $\mathcal{M}_{t,t}$ and from Model $\mathcal{M}_{tw,\emptyset}$ to Model $\mathcal{M}_{t,\emptyset}$. This reinforces the notion that particular head words have structural preferences, so that making the P_M parameters dependent on head words would capture such preferences. As for effects involving dependence on the head tag t_h , observe that when moving from Model $\mathcal{M}_{tw,t}$ to Model $\mathcal{M}_{tw,\emptyset}$, there is a small drop in both recall and precision, whereas when making an analogous move from Model $\mathcal{M}_{t,t}$ to Model $\mathcal{M}_{t,\emptyset}$, there is a drop in recall, but a slight *gain* in precision (the two moves are analogous, in that in both cases, t_h is dropped from the context of P_{M_w}). It is not evident why these two moves do not produce similar performance losses, but in both cases, the performance drops are small relative to those observed when eliminating w_h from the conditioning contexts, indicating that head words matter far more than parts of speech for determining structural preferences, as one would expect.

Briefly noted: unlexicalized parsing

Model $\mathcal{M}_{\emptyset,\emptyset}$ is a model in which lexicalized modifiers are generated with no dependence on the head word or its tag, but this model should not be mistaken for an unlexicalized parsing model. Unlexicalized parsing models—or, more accurately, “somewhat” lexicalized parsing models—have recently been explored by Klein and Manning (2003). In an effort to get a similar unlexicalized model from the Collins model as a baseline, we used the modified P_M schemata from $\mathcal{M}_{\emptyset,\emptyset}$ (partially-lexicalized modifiers were generated with no dependence whatsoever on the head child’s head word and tag) and we seriously crippled the P_{M_w} parameter class to deliver estimates with two levels of back-off using very little context: $\lambda_0 \hat{p}(w_{M_i} | M_i, t_{M_i}) + (1 - \lambda_0) \hat{p}(w_{M_i} | t_{M_i})$. In this way, head words of lexicalized modifier nodes were generated conditioning on nothing more than the (unlexicalized) modifier label and the head word’s part-of-speech tag. Table 6.4 presents the results of this experiment alongside the results of (Klein and Manning, 2003) and (Magerman, 1995). While Klein and Manning (2003) accurately describe their results as superior to early lexicalized parsing models such as Magerman’s (1994), this new baseline puts their results in the proper perspective relative to a modern lexicalized model that has been appropriately hobbled. For example, the F-measure

Model	LR	LP	CB	0 CBs	F
≤ 40 words					
Klein and Manning	85.7	86.9	1.10	60.3	86.3
Unlex Collins	84.7	84.7	1.24	56.8	84.7
Magerman (1995)	84.6	84.9	1.26	56.6	84.7
≤ 100 words					
Klein and Manning	85.1	86.3	1.31	57.2	85.7
Unlex Collins	83.9	84.1	1.46	54.0	84.0

Table 6.4: Results on §23 with the “unlexicalized” version of Collins’ Model 2 and the results of Klein and Manning (2003) and Magerman (1995).

of the hobbled Collins model is identical to that of Magerman’s lexicalized model on sentences of length ≤ 40 words.

6.2 Recasting a generative parsing model

It is tempting to refer to the parameter classes of a generative model using only their maximal-context histories, thinking that back-off is used simply to make these distributions smoother. However, we have seen, both in Chapter 3 and in §6.1, that Collins’ parsing model does not necessarily make very much use of its maximal-context distributions. So, we would recast an overall model of this type as a collection of mixture models, the number of which is determined by the number of instantiations of the parameter classes in the observed (training) data, where the relative importance of the mixture components—back-off levels—for a given model is necessarily an empirical question. This point will become even clearer in Chapter 7.

6.3 Arguments and heads: using EM to reestimate the model

Much of the material in this section is from (Chiang and Bikel, 2002), which presented ways to aid parser developers in recovering the latent information on which most of the recent, lexicalized parsers rely. The EM experiments reported in this section were performed using Chiang’s stochastic tree-insertion grammar model. The EM method employed for that research had a flaw, and a

solution is presented in Section 6.4, along with experiments with our parsing engine using the new method.

6.3.1 Motivation

Most work in statistical parsing does not operate in the realm of parse trees as they appear in many treebanks, but rather on trees transformed via augmentation of their node labels (such as discussed in §3.3), or some other transformation (Johnson, 1998). This methodology is illustrated in Figure 6.1. The information included in the node labels' augmentations may include lexical items, or a node label suffix to indicate the node is an argument and not an adjunct; such extra information may be viewed as latent information, in that it is not directly present in the treebank parse trees, but may be recovered by some means. The process of recovering this latent information has largely been limited to the hand-construction of heuristics. However, as is often the case, hand-constructed heuristics may not be optimal or very robust. Also, the effort required to construct such rules can be considerable. In both respects, the use of such rules runs counter to the data-driven approach to statistical parsing. Magerman was acutely aware of this last point: in Chapter 9 of his thesis (1994), he describes how one of his aims had been that “manual knowledge engineering should be kept at a minimum”. He goes on to point out that the most blatant violation of this principle was the use of the hand-constructed head-finding table. Ideally, Magerman claims, distinguishing a child of a parent node as a head should be determined automatically, by encoding head-selection as yet another hidden component of the overall model. However, such a scheme, he notes, would explode the already-large search space.

Partly to this end, we explore a novel use of Expectation-Maximization (Dempster, Laird, and Rubin, 1977) that iteratively reestimates a parsing model using the augmenting heuristics as a starting point. Specifically, the EM algorithm we use is a variant of the Inside-Outside algorithm (Baker, 1979; Lari and Young, 1990; Hwa, 1998). The reestimation adjusts the model's parameters in the augmented parse-tree space to maximize the likelihood of the observed (incomplete) data, in the hopes of finding a better distribution over augmented parse trees (the complete data). The ultimate goal of this work is to minimize the human effort needed when adapting a parsing model

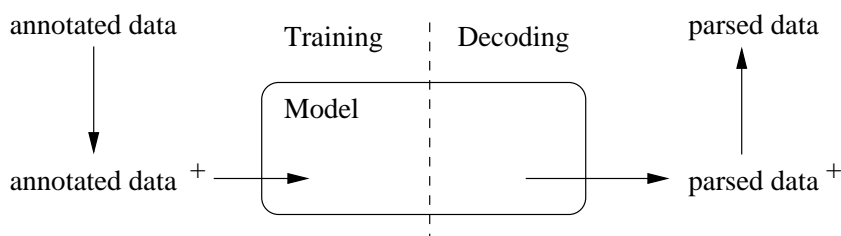


Figure 6.1: Methodology for the development of a statistical parser. A + indicates augmentation.

to a new domain.

6.3.2 Background

6.3.2.1 Head-lexicalization

As we have seen, many of the recent, successful statistical parsers have made use of lexical information or an implicit lexicalized grammar, both for English and, more recently, for other languages. All of these parsers recover the “hidden” lexicalizations in a treebank and find the most probable lexicalized tree when parsing, only to strip out this hidden information prior to evaluation. Also, in all these parsing efforts lexicalization has meant finding heads of constituents and then propagating those lexical heads to their respective parents. In fact, nearly identical head-lexicalizations were used in the discriminative models described in (Magerman, 1995; Ratnaparkhi, 1997), the lexicalized PCFG models in (Collins, 1999), the lexicalized TAG extractor in (Xia, 1999) and the stochastic lexicalized TAG models in (Chiang, 2000; Sarkar, 2001; Chen and Vijay-Shanker, 2000). Inducing a lexicalized structure based on heads has a two-pronged effect: it not only allows statistical parsers to be sensitive to lexical information by including this information in the probability model’s dependencies, but it also determines which of all possible dependencies—both syntactic and lexical—will be included in the model itself. For example, in Figure 6.2, the nonterminal NP(NN–boy) is dependent on VP(VBD–caught) and not the other way around.

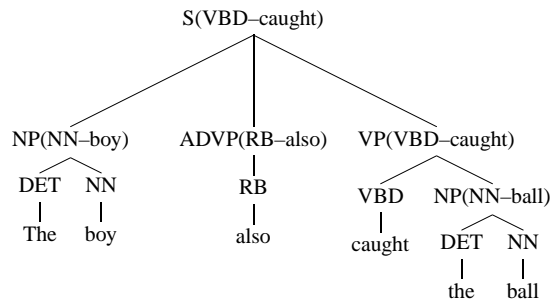


Figure 6.2: A simple lexicalized parse tree.

6.3.2.2 Other tree transformations

Lexicalization via head-finding is but one of many possible tree transformations that might be useful for parsing. As explored thoroughly by Johnson (1998), even simple, local syntactic transformations on training trees for an unlexicalized PCFG model can have a significant impact on parsing performance. In a similar vein, we saw in Chapter 3 how Collins (1999) devises rules to identify arguments, *i.e.*, constituents that are required to exist on a particular side of a head child constituent dominated by a particular parent, allowing the model to incorporate a type of subcategorization information. While the model is augmented to include this subcat-prediction feature, the actual identification of arguments is performed as one of many preprocessing steps on training trees, using a set of rules similar to those used for the identification of heads. Also, we have seen how (Collins, 1999) makes use of several other transformations, such as the identification of subjectless sentences (augmenting S nodes to become SG) and the augmentation of nonterminals for gap threading. Xia (1999) combines head-finding with argument identification to extract elementary trees for use in the lexicalized TAG formalism. Other researchers investigated this type of extraction to construct stochastic TAG parsers (Chiang, 2000; Chen and Vijay-Shanker, 2000; Sarkar, 2001).

6.3.2.3 Problems with heuristics

While head-lexicalization and other tree transformations allow the construction of parsing models with more data-sensitivity and richer representations, crafting rules for these transformations has

been largely an art, with heuristics handed down from researcher to researcher. What's more, on top of the large undertaking of designing and implementing a statistical parsing model, the use of heuristics has required a further effort, forcing the researcher to bring both linguistic intuition and, more often, engineering savvy to bear whenever moving to a new treebank. For example, in the rule sets used by the parsers described in (Magerman, 1995; Ratnaparkhi, 1997; Collins, 1999), the sets of rules for finding the heads of ADJP, ADVP, NAC, PP and WHPP include rules for picking either the rightmost or leftmost FW (foreign word). The apparently haphazard placement of these rules that pick out FW and the rarity of FW nodes in the data strongly suggest these rules are the result of engineering effort. Furthermore, it is not at all apparent that tree-transforming heuristics that are useful for one parsing model will be useful for another. Finally, as is often the case with heuristics, those used in statistical parsers tend not to be data-sensitive, and ironically do not rely on the words themselves.

6.3.3 Rule-based augmentation

In the interest of reducing the effort required to construct augmentation heuristics, we would like a notation for specifying rules for selecting nodes in bracketed data that is both flexible enough to encode the kinds of rule sets used by existing parsers, and intuitive enough that a rule set for a new language can be written easily without knowledge of computer programming. Such a notation would simplify the task of writing new rule sets, and facilitate experimentation with different rules. Moreover, rules written in this notation would be interchangeable between different models, so that, ideally, adaptation of a model to a new corpus would be trivial.

We refer the reader to (Chiang and Bikel, 2002) for the details on our notation, and simply mention that using our new, fairly simple syntax, one can fully encode the head/argument rules used by Xia (1999), and the family of rule sets used by Black, Magerman, Collins, Ratnaparkhi, and others.

6.3.4 Unsupervised learning of augmentations

In the type of approach we have been discussing so far, hand-written rules are used to augment the training data, and this augmented training data is then used to train a statistical model. However, if we train the model by maximum-likelihood estimation, the estimate we get will indeed maximize the likelihood of the training data as augmented by the hand-written rules, but not necessarily that of the training data itself. In this section we explore the possibility of training a model directly on unaugmented data.

A generative model that estimates $P(S, T, T^+)$ (where T^+ is an augmented tree) is normally used for parsing, by computing the most likely (T, T^+) for a given S . But we may also use it for augmenting trees, by computing the most likely T^+ for a given sentence-tree pair (S, T) . From the latter perspective, because its trees are unaugmented, a treebank is a corpus of incomplete data, warranting the use of unsupervised learning methods to reestimate a model that includes hidden parameters. The approach we take below is to seed a parsing model using hand-written rules, and then use the Inside-Outside algorithm to reestimate its parameters. The resulting model, which locally maximizes the likelihood of the *unaugmented* training data, can then be used in two ways: one might hope that as a parser, it would parse more accurately than a model which only maximizes the likelihood of training data augmented by hand-written rules; and that as a tree-augmenter, it would augment trees in a more data-sensitive way than hand-written rules.

6.3.4.1 Background: tree adjoining grammar

The parsing model we use is based on the stochastic tree-insertion grammar (TIG) model described by Chiang (2000). TIG (Schabes and Waters, 1995) is a weakly-context free restriction of tree adjoining grammar (Joshi and Schabes, 1997), in which tree fragments called *elementary trees* are combined by two composition operations, *substitution* and *adjunction* (see Figure 6.3). In TIG there are certain restrictions on the adjunction operation. Chiang’s model adds a third composition operation called *sister-adjunction* (see Figure 6.3), borrowed from D-tree substitution grammar (Rambow, Vijay-Shanker, and Weir, 1995).³

³The parameters for sister-adjunction in the present model differ slightly from the original. In the original model, all the modifier auxiliary trees that sister-adjoined at a particular position were generated independently, except that each

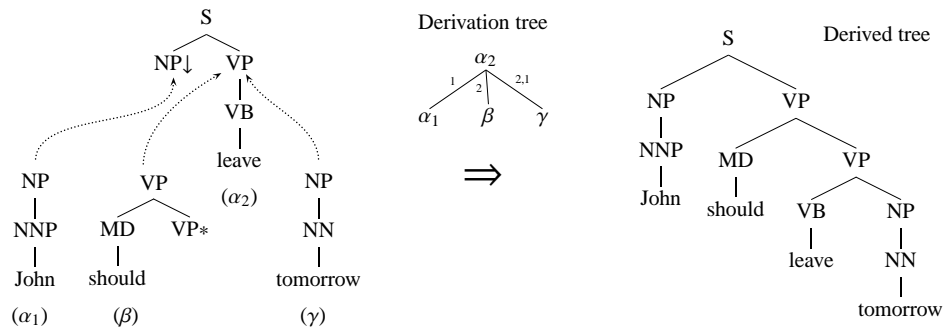


Figure 6.3: Grammar and derivation for “John should leave tomorrow.” In this derivation, α_1 gets substituted, β gets adjoined, and γ gets sister-adjoined.

There is an important distinction between *derived trees* and *derivation trees* (see Figure 6.3). A derivation tree records the operations that are used to combine elementary trees into a derived tree. Thus there is a many-to-one relationship between derivation trees and derived trees: every derivation tree specifies a derived tree, but a derived tree can be the result of several different derivations.

The model can be trained directly on TIG derivations if they are available, but corpora like the Penn Treebank have only derived trees. Just as Collins uses rules to identify heads and arguments and thereby lexicalize trees, Chiang uses nearly the same rules to reconstruct derivations: each training example is broken into elementary trees, with each head child remaining attached to its parent, each argument broken into a substitution node and an initial root, and each adjunct broken off as a modifier auxiliary tree.

However, in this experiment we view the derived trees in the Treebank as incomplete data, and try to reconstruct the derivations (the complete data) using the Inside-Outside algorithm.

6.3.4.2 Implementation

The expectation step (E-step) of the Inside-Outside algorithm is performed by a parser that computes all possible derivations for each parse tree in the training data. It then computes inside and

sister-adjunction was conditioned on whether it was the first at that position. In the present model, each sister-adjunction is conditioned on the root label of the previous modifier tree.

outside probabilities as in Hwa’s experiment (1998), and uses these to compute the expected number of times each event occurred. For the maximization step (M-step), we obtain a maximum-likelihood estimate of the parameters of the model using relative-frequency estimation, just as in the original experiment, as if the expected values for the complete data were the training data.

Smoothing presents a special problem. Using the same notation as §3.5.7, let ϕ_i be a function from a full history context Y to some less-specific context, and let e_i be the estimate of $p(X | \phi_i(Y))$ for some future X . Estimates are smoothed via deleted interpolation as described in §3.5.7, with a smoothing factor of 5.0 and a smoothing term of 0.0. There is also a second factor, following the formula from (Bikel et al., 1997):

$$\lambda_i = \left(1 - \frac{c_{i-1}}{c_i}\right) \left(\frac{c_i}{c_i + 5u_i}\right) \quad (6.1)$$

where c_i is the number of occurrences in training of the context $\phi_i(Y)$ (and $c_0 = 0$), and u_i is the diversity of that context as seen in training. The additional factor of $1 - \frac{c_{i-1}}{c_i}$ essentially “skips” the estimate at level i if the context at level $i - 1$ had roughly the same many occurrences in the training, preferring the estimate delivered by the more-detailed model (see (Bikel et al., 1997) for more details on this additional factor).

There are several ways one might incorporate this smoothing into the reestimation process, and we chose to depart as little as possible from the original smoothing method: in the E-step, we use the smoothed model, and after the M-step, we use the original formula (6.1) to recompute the smoothing weights based on the new counts computed from the E-step. While simple, this approach has two important consequences. First, since the formula for the smoothing weights intentionally does *not* maximize the likelihood of the training data, each iteration of reestimation is not guaranteed to increase the likelihood of the training data. Second, reestimation tends to increase the size of the model in memory, since smoothing gives nonzero expected counts to many events which were unseen in training. Therefore, since the resulting model is quite large, if an event at a particular point in the derivation forest has an expected count below 10^{-15} , we throw it out.

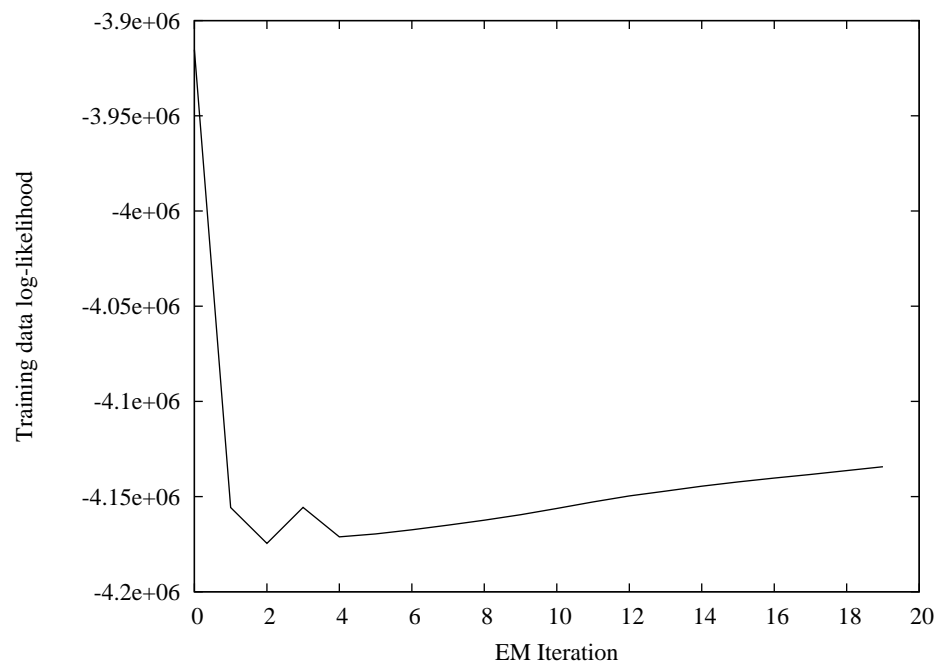


Figure 6.4: Log-likelihood of the training data after each of the EM iterations using the full head-finding rule set for the initial model.

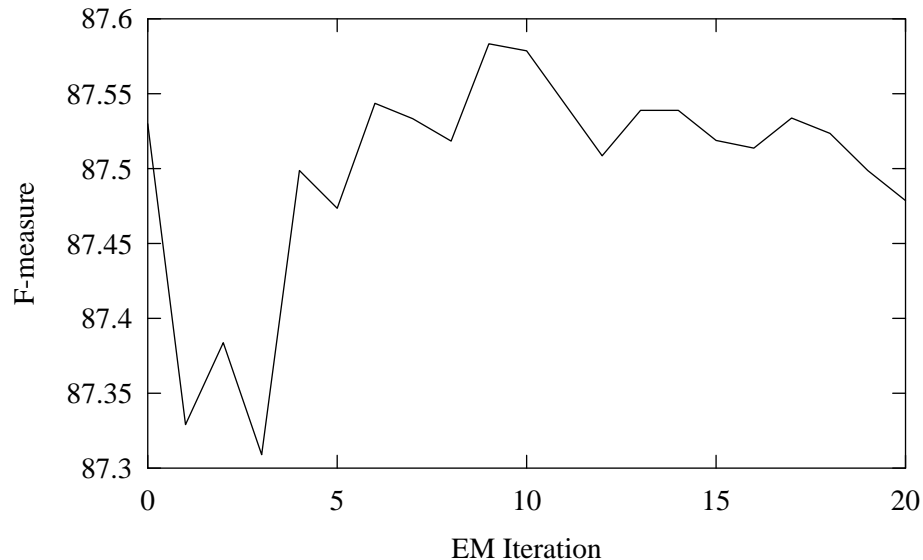


Figure 6.5: English, starting with full rule set

6.3.4.3 Experiment

We first trained the initial model on sections 02–21 of the WSJ corpus using the original head rules, and then ran the Inside-Outside algorithm on the same data. We tested each successive model on some held-out data (section 00), using a beam width of 10^4 , to determine at which iteration to stop. The log-likelihood of the training data is shown in Figure 6.4, and the F-measure (harmonic mean of labeled precision and recall) for sentences of length ≤ 100 for each iteration is shown in Figure 6.5. We then selected the ninth reestimated model and compared it with the initial model on section 23 (see Figure 6.5). This model did only marginally better than the initial model on section 00, but it actually performs worse than the initial model on section 23. One explanation is that the head rules, since they have been extensively fine-tuned, do not leave much room for improvement. To test this, we ran two more experiments.

The second experiment started with a simplified rule set, which simply chooses either the leftmost or rightmost child of each node as the head, depending on the label of the parent: *e.g.*, for VP, the leftmost child is chosen; for NP, the rightmost child is chosen. The argument rules, however,

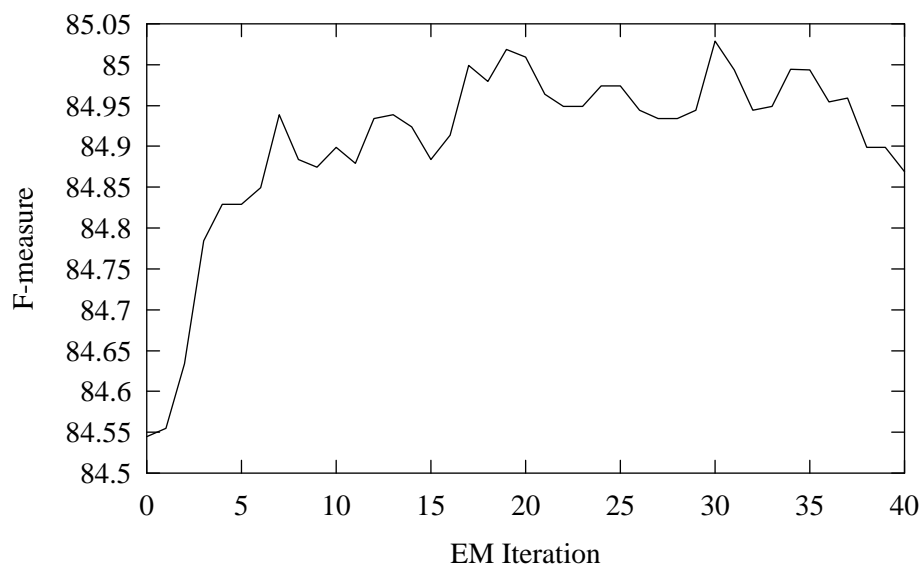


Figure 6.6: English, starting with simplified rule set

were not changed. This rule set is supposed to represent the kind of rule set that someone with basic familiarity with English syntax might write down in a few minutes. The reestimated models seemed to improve on this simplified rule set when parsing section 00 (see Figure 6.6); however, when we compared the 30th reestimated model with the initial model on section 23 (see Table 6.5), there was no improvement.

The third experiment was on the Chinese Treebank, starting with the same head rules used in (Bikel and Chiang, 2000). These rules were originally written by Xia for grammar development, and although we have modified them for parsing, they have not received as much fine-tuning as the English rules have. We trained the model on sections 001–270 of the Penn Chinese Treebank, and reestimated it on the same data, testing it at each iteration on sections 301–325 (Figure 6.7). We selected the 38th reestimated model for comparison with the initial model on sections 271–300 (Table 6.5). Here we did observe a small improvement: an error reduction of 3.4% in the F-measure for sentences of length ≤ 40 .

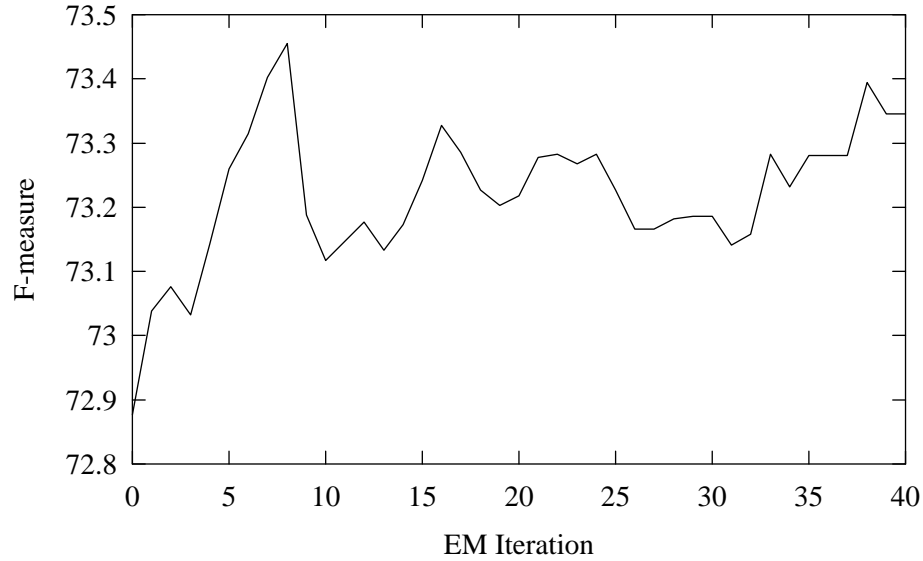


Figure 6.7: Chinese, starting with full rule set.

Model	Step	≤ 100 words					≤ 40 words				
		LR	LP	CB	0CB	≤ 2 CB	LR	LP	CB	0CB	≤ 2 CB
Original	initial	86.95	87.02	1.21	62.38	82.33	87.68	87.76	1.02	65.30	84.86
Original	9	86.37	86.71	1.26	61.42	81.79	87.18	87.48	1.06	64.41	84.23
Simple	initial	84.50	84.18	1.54	57.57	78.35	85.46	85.17	1.29	60.71	81.11
Simple	30	84.21	84.50	1.53	57.95	77.77	85.12	85.35	1.30	60.94	80.62
Chinese	initial	75.30	76.77	2.72	45.95	67.05	78.37	80.03	1.79	52.82	74.75
Chinese	38	75.20	77.99	2.66	47.69	67.63	78.79	81.06	1.69	54.15	75.08

Table 6.5: Results on blind test data. Original = trained on English with original rule set; Simple = English, simplified rule set. LR = labeled recall, LP = labeled precision; CB = average crossing brackets, 0 CB = no crossing brackets, ≤ 2 CB = two or fewer crossing brackets. All figures except CB are percentages.

6.4 True EM with a smoothed model

As mentioned in the previous section, smoothing presents special problems when trying to reestimate a model using EM. The first problem is that the E-step produces fractional counts, and many smoothing techniques rely on statistics based on integer counts (Goodman (2001) mentions fractional counts as an area for future smoothing research). With Witten-Bell specifically, the technique computes a proxy for the perplexity of the distribution with which it is associated (as we saw in Chapter 3, footnote 27) by computing the quantity

$$\frac{\bar{n}}{(\bar{n} + f_f)}, \tag{6.2}$$

where f_f is the smoothing factor (5 is used above) and \bar{n} is the average number of outcomes per future. This quantity is close to its maximum of 1 with a “spikey” distribution (because \bar{n} will be large), and is at its lowest when the distribution is uniform, with each observed future occurring exactly once.

Witten-Bell and entropy

To see the mathematical relationship between the Witten-Bell smoothing value associated with a particular history context and the entropy for the conditional distribution with that history context, observe the following. Let the $\hat{p}(\cdot|B)$ be the (estimated) conditional distribution for which a smoothing value is to be computed, and let c be the count function, such that $c(B)$ is the number of times the history context B occurred in training, and $c(A, B)$ is the number of times B occurred with some future A . The maximum likelihood estimate $\hat{p}(A|B) = \frac{c(A,B)}{c(B)}$. Let u be the *diversity* of the distribution $\hat{p}(\cdot|B)$ (the number of unique futures observed with B). Note that \bar{n} is the unweighted average of $c(A, B)$:

$$\bar{n} = \frac{c(B)}{u} = \sum_A \frac{1}{u} c(A, B). \tag{6.3}$$

This value is closely related to the expected value for $c(A, B)$, which is the following weighted average:

$$\begin{aligned} E_{\hat{p}(\cdot|B)}c(A, B) &= \sum_A \hat{p}(A|B) \cdot c(A, B) \\ &= \sum_A \hat{p}(A|B) \cdot \hat{p}(A|B) \cdot c(B) \\ &= c(B) \cdot E_{\hat{p}(\cdot|B)}\hat{p}(A|B). \end{aligned} \tag{6.4}$$

Now, consider that the entropy of $\hat{p}(\cdot|B)$ is simply the expected log of the inverse of $\hat{p}(A|B)$ for all possible A :

$$H(\hat{p}(\cdot|B)) = - \sum_A \hat{p}(A|B) \lg \hat{p}(A|B) = E_{\hat{p}(\cdot|B)} \lg \frac{1}{\hat{p}(A|B)}. \tag{6.5}$$

This is the expected number of bits one would need in an optimal encoding of samples from the distribution. The perplexity of the distribution is $2^{H(\hat{p}(\cdot|B))}$, and may be interpreted as the expected number of outcomes for a random sample. Relating (6.4) to perplexity, note that

$$2^{-H(\hat{p}(\cdot|B))} = \prod_A \hat{p}(A|B)^{\hat{p}(A|B)} \leq E_{\hat{p}(\cdot|B)}\hat{p}(A|B) \tag{6.6}$$

since a geometric mean is always less than or equal to an arithmetic mean (Hardy, Littlewood, and Pólya, 1988). Multiplying both sides of (6.6) by $c(B)$ we have

$$c(B) \cdot 2^{-H(\hat{p}(\cdot|B))} \leq c(B) \cdot E_{\hat{p}(\cdot|B)}\hat{p}(A|B). \tag{6.7}$$

or

$$\frac{c(B)}{2^{H(\hat{p}(\cdot|B))}} \leq E_{\hat{p}(\cdot|B)}c(A, B) \tag{6.8}$$

Since u is the *total* number of possible outcomes, $u \geq 2^{H(\hat{p}(\cdot|B))}$, and therefore

$$\bar{n} = \frac{c(B)}{u} \leq \frac{c(B)}{2^{H(\hat{p}(\cdot|B))}}. \tag{6.9}$$

Thus,

$$\bar{n} \leq c(B) \cdot 2^{-H(\hat{p}(\cdot|B))} \leq E_{\hat{p}(\cdot|B)}c(A, B) \tag{6.10}$$

which is to say that the unweighted average of $c(A, B)$, \bar{n} , bounds $c(B) \cdot 2^{-H(\hat{p}(\cdot|B))}$ from below and the weighted average $E_{\hat{p}(\cdot|B)}c(A, B)$ bounds $c(B) \cdot 2^{-H(\hat{p}(\cdot|B))}$ from above. So to the extent that \bar{n}

is an approximation of $c(B) \cdot 2^{-H(\hat{p}(\cdot|B))}$, it is computing a value related to the entropy/perplexity of the distribution with which it is associated. In Witten-Bell, the quantity \bar{n} is transformed in a somewhat ad-hoc way to have a range in the $[0, 1)$ interval, as we have seen above in Equation 6.2 (and when the Witten-Bell formula of Equation 6.2 yields a non-zero value, the range is $[\frac{1}{1+f_f}, 1)$ since \bar{n} cannot be less than 1). Crucially, the transformed function is still at its minimum when \bar{n} is at its minimum value of 1, and so the transformed function is still effectively measuring the closeness of the observed distribution to uniform. However, it has the additional, desirable property of giving less weight to uniform distributions with low counts for all possible events.

Fractional counts

With fractional counts, however, the quantity \bar{n} no longer has this meaning, nor is it entirely clear how or if the formula can be modified while keeping its semantics as a confidence estimate and simultaneously preserving its simplicity of computation. Also, we would like to remedy the situation where likelihood of the training data is not guaranteed to rise (since we did not keep this guarantee in our initial experiments reported in the previous section, it is more accurate to say that we used an *approximation* to EM).

Our solution to both of these problems is straightforward to describe: all smoothing weights are computed as normal with the initial model, and then are saved and used *throughout the EM training procedure*. This solution has two important consequences: it not only preserves the EM guarantee of the training data likelihood increasing monotonically, but also keeps the number of parameters in the model constant over all iterations. The first consequence is fairly easy to see, but the second is slightly more involved.

Smoothing weights are normally “hidden” data, in that they are not normally directly observable, and, in fact, EM can be used to compute smoothing weights for many different types of models. In our situation, though, we are not interested in reestimating that part of the model. So, by computing these weights once and for all, they now become part of the of the observed data, as far as the EM procedure is concerned, and so crucially the E-step occurs *with respect to the fixed smoothing weights*. Thus, all EM guarantees are kept.

The second consequence of our solution, keeping model size constant, is a result of the way this smoothing procedure handles unseen history events. As we saw in §3.5.7, the formula for computing smoothing weights is not precisely as we have shown it above in Equation 6.1. In fact, it is as in Equation 3.19, repeated here:

$$\lambda_i = \begin{cases} \frac{c_i}{c_i + f_i + f_j u_i} & \text{if } c_i > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (6.11)$$

Crucially, note the second case in the equation, which states that smoothing weights are 0 for all unseen history events. Even though this is an interpolated smoothing technique, in the case of an unseen history context, the model truly “backs off” to the $i + 1$ context in the back-off chain.⁴ To illustrate what happens with such a technique and EM, consider the following small example. Suppose we have an n -gram language model that computes

$$p(w_i | w_{i-1}, w_{i-2}) = \lambda_0 p(w_i | w_{i-1}, w_{i-2}) + (1 - \lambda_0) p(w_i | w_{i-1}), \quad (6.12)$$

where smoothing weights are computed using Witten-Bell. Since

$$p(w_i | w_{i-1}) = \sum_{w_{i-2}} p(w_i | w_{i-1}, w_{i-2}), \quad (6.13)$$

we derive estimates for the bigram model from trigram events. Now suppose that we are doing EM on some held-out data, and are computing the expected counts of the word sequence `John's toy ball`. Let's say that the bigram `toy ball` occurred in the original training data and that $\hat{p}(\text{ball}|\text{toy}) = 0.1$, but that the trigram `John's toy ball` never occurred. In this case, $\lambda_0 = 0$. Nevertheless, the trigram `John's toy ball` has a non-zero expected count, which will be $(1 - 0) \cdot \hat{p}(\text{ball}|\text{toy}) = 0.1$, and so our expectation counter will output the event `John's toy ball` with this fractional count, and we will have added a new parameter, $p(\cdot|\text{toy}, \text{John's})$, to our model.

With our “proper EM” solution, however, all smoothing weights are computed on the initial training data and are left unchanged throughout EM, *including those weights that were zero*. Therefore, during the M-step, fractional counts for “unseen” events are only used to produce updated estimates for seen events. In our example above, $\hat{p}(\text{ball}|\text{toy}, \text{John's}) = 0.1$ will only be used when

⁴We are making a distinction between interpolation and backing off here. In fact, many of the “interpolated” techniques described in Goodman (2001) do true back-off in the face of unseen histories.

deriving an updated estimate for $\hat{p}(\text{ball}|\text{toy})$ via the summation of Equation 6.13. Put another way, during the M-step only the counters associated with the back-off levels that caused the new event to be “hallucinated” in the first place are updated; in the above example, the counter associated with the bigram `toy ball` is updated while no new counter is created for `John’s toy ball`.⁵ So, just as with the E-step, the M-step also occurs *with respect to the fixed smoothing weights*.

6.4.1 Experiments Redux

We duplicated the experiments of §6.3, but this time using the true EM method described above and the core Collins model. Unlike the full-blown “Collins-emulation” model for English, we adjusted settings so that the model was not deficient in either its handling of unknown words or in its method of smoothing.

As before, the first experiment trains the initial model on Sections 02–21 of the Penn Treebank using the standard, “full” set of head-finding rules, and then performs EM treating the unlexicalized, unaugmented trees as the observed data. Figure 6.8 shows the likelihood of the training after each of the 40 EM iterations. The monotonicity of this graph is in stark contrast to the approximate EM method of the previous section, which consistently produces a sharp dip in likelihood after the first iteration before an eventual monotonic rise, as shown in Figure 6.4. Figure 6.9 shows the results of testing the English model, where heads had been determined for the initial model using the standard, fine-tuned head-finding rule set; the test set is our standard development test set, Section 00 of the Penn Treebank. The peak in this graph is for the model after the 17th iteration, so we also tested both the initial and the 17th iteration model on the blind test set, Section 23, but there was only a statistically insignificant improvement in F-measure from the initial to the 17th model, as shown in Table 6.6 (“Original”).

For the second experiment, we used the same simplified English head-finding rule set as in the previous section, but with the probabilistically-sound version of our Collins-style baseline model. Figure 6.10 shows the increase in likelihood of the training data, and Figure 6.11 shows the evaluation on the development test set, Section 00. The peak in this graph is for the model after iteration

⁵This is obviously a toy example, as we have no real hidden parameters, and so the updated estimates in our language model will not change from iteration to iteration.

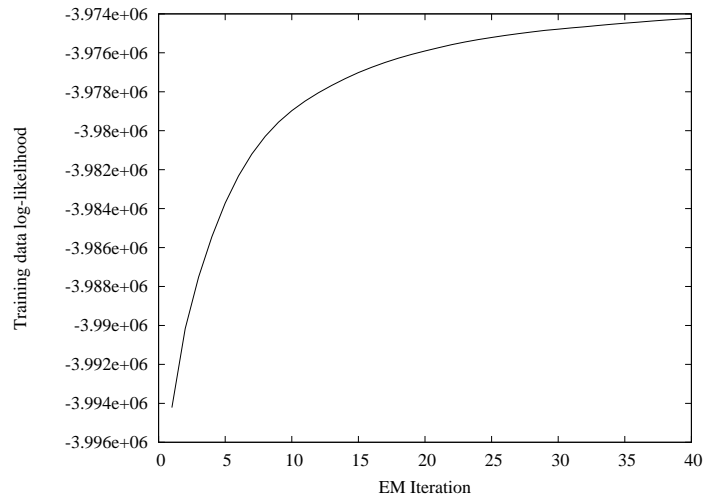


Figure 6.8: Log-likelihood of the training data after each of the EM iterations using the full head-finding rule set for the initial model.

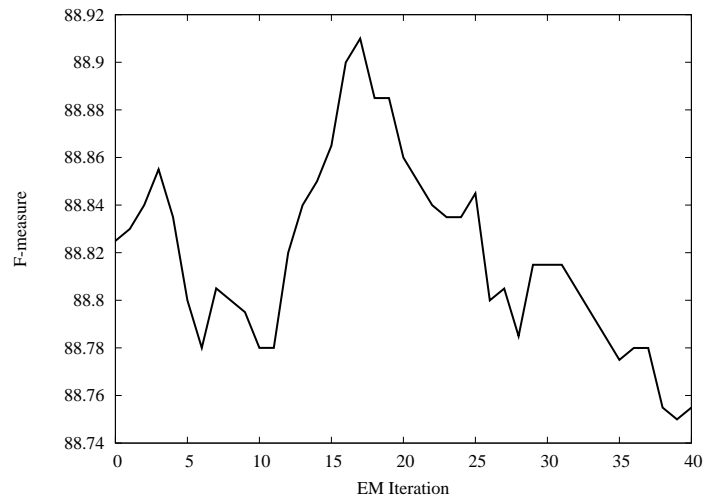


Figure 6.9: F-measure for all sentences of Section 00 with EM models, using full head-finding rule set for the initial model.

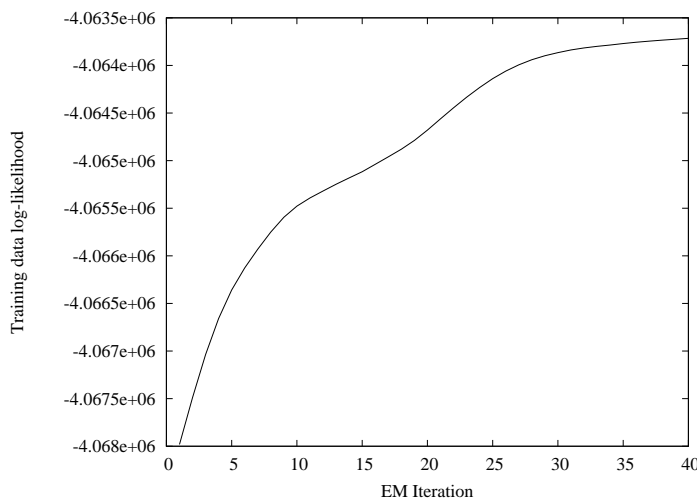


Figure 6.10: Log-likelihood of the training data after each of the EM iterations using the simplified head-finding rule set for the initial model.

8; as before, there is a measurable but statistically-insignificant improvement on the blind test (Table 6.6, “Simple”).

The third experiment was on the Chinese Treebank v1.0 (CTB1.0), using the full head-finding rule set for the initial model. Figure 6.12 shows the log-likelihood of the training data, and Figure 6.13 shows the evaluation on the development test set, composed of Sections 301–325. The model produced after iteration 33 has the highest F-measure on this test set; unfortunately, as with most of these experiments, the results on the blind test set (Sections 271–300) were no better than with the initial model (in fact, they were measurably worse, but well within the margin of error; see Table 6.6, “Chinese”).

6.4.2 Random head-finding

One of the issues with the experiments of the previous sections is that they all operate within the narrow ambiguity allowed by the set of head-finding heuristics used for the initial model. For example, if a head-finding rule is of the form “if the parent is VP, pick the leftmost one of {VB, VBP, VBD, VBZ, VBG}, else pick the rightmost VP”, then a model trained on data augmented

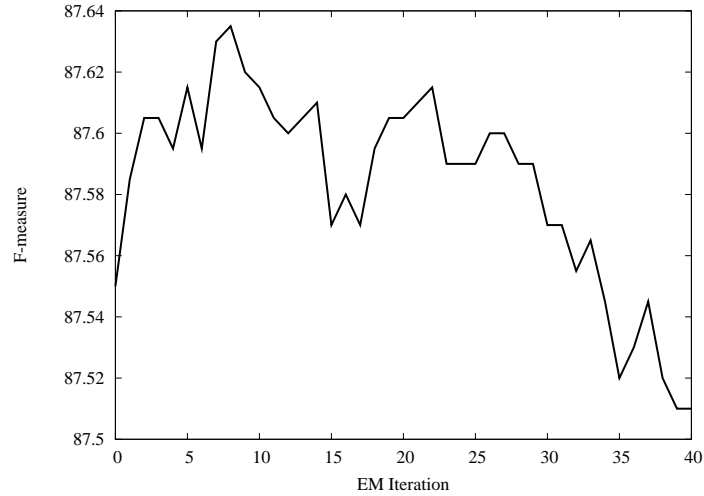


Figure 6.11: F-measure for all sentences of Section 00 with EM models, using simplified head-finding rule set for the initial model.

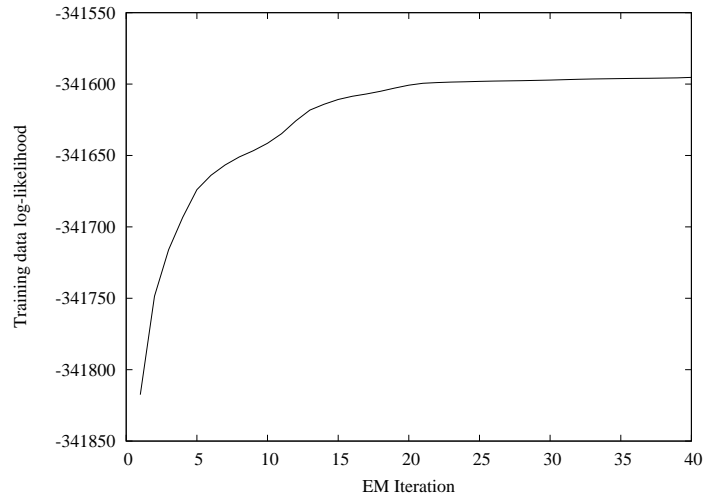


Figure 6.12: Log-likelihood of the CTB1.0 training data after each of the EM iterations on the Chinese model.

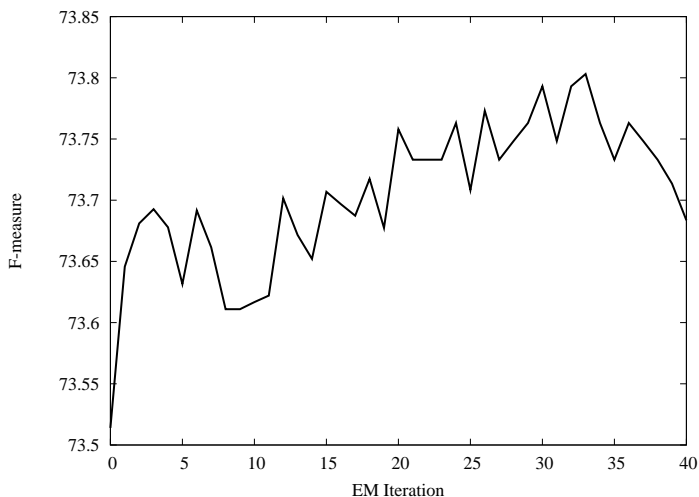


Figure 6.13: F-measure for all sentences of the CTB1.0 development test set with EM models.

Model	Step	≤ 100 words					≤ 40 words				
		LR	LP	CB	OCB	≤ 2 CB	LR	LP	CB	OCB	≤ 2 CB
Original	initial	88.16	88.30	1.05	64.79	85.29	88.68	88.82	0.89	67.41	87.29
Original	17	88.22	88.32	1.05	63.96	85.42	88.73	88.80	0.90	66.43	87.38
Simple	initial	86.86	87.05	1.16	62.59	83.55	87.45	87.66	0.98	65.18	85.73
Simple	8	86.89	87.04	1.16	62.72	83.51	87.51	87.65	0.98	65.40	85.87
Chinese	initial	73.84	78.31	2.50	47.41	69.83	76.94	80.52	1.64	53.49	76.41
Chinese	33	73.76	78.16	2.51	46.84	69.54	76.87	80.33	1.65	52.82	75.75
Random	initial	86.33	86.47	1.24	59.68	82.68	86.90	87.04	1.05	62.27	85.10
Random	35	86.22	86.34	1.25	59.80	82.64	86.80	86.93	1.07	62.49	84.88

Table 6.6: EM results on blind test data (English: §23 of the Penn Treebank, CTB1.0: §§271–300).

with this rule will *only* allow heads of VPs that are one of $\{VP, VB, VBP, VBD, VBZ, VBG\}$.⁶ By contrast, the decoder, modulo pruning, searches the entire space of head lexicalizations, and is limited only by where the model imposes a constraint via a zero probability estimate.

As a way to allow the EM method to search a larger space of head lexicalizations, we added a run-time option to our head-finder that allows the user to specify a probability that a head child will be chosen at random. That is, as each training tree is broken into a series of CFG productions, with some probability p the head child on the right-hand side of a production will be chosen randomly, and with probability $1 - p$ the head child will be chosen according to the heuristics.

We performed experiments with several different values of p . Figure 6.14 shows the likelihood of the training data when $p = 0.05$, and Figure 6.15 shows the evaluation on the development test set, Section 00. Because random head-finding allows so many more parameters into the model, for efficiency reasons we used a reduced training data set of Sections 02–06 of the Penn Treebank, which is roughly one-quarter the size of the standard training data set (Sections 02–21). As with the other EM experiments, evaluation on blind test data revealed a measurable but statistically insignificant difference (see Table 6.6, “Random”).

6.4.3 Discussion of results

It would appear that, for both English and Chinese, whether using simple or fine-tuned head rules, the accuracy gains due to EM, if any, are vanishingly small, and thus are probably not worth the extra time of the training method. Nevertheless, one cannot deny that (1) likelihood of the training is increasing substantially in all cases and (2) in nearly all cases accuracy measurably improves before decreasing again in the characteristic overfitting scenario. While any gains in accuracy are *measurable*, they are not statistically *significant*, so one might ask, “What is changing about the model that so dramatically increases the likelihood of the training while barely affecting the standard accuracy metrics?”

⁶This is not quite the case, for in the real head-finding system, a heuristic must always provide a head for a given parent. A heuristic for a given parent may provide a default of rule the form “pick the leftmost/rightmost child”; for heuristics without defaults, a global default rule is used. Nevertheless, even with this system and with typical real-world training data, the set of head nonterminal labels for a given parent is generally much smaller than the full set of nonterminals.

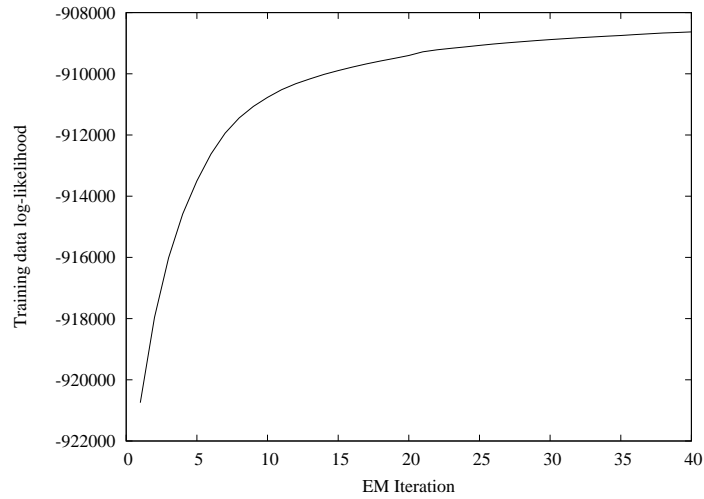


Figure 6.14: Log-likelihood of reduced-size training set (Penn Treebank §§02–06) after each of the EM iterations, using a 5% random head-finder for the initial model.

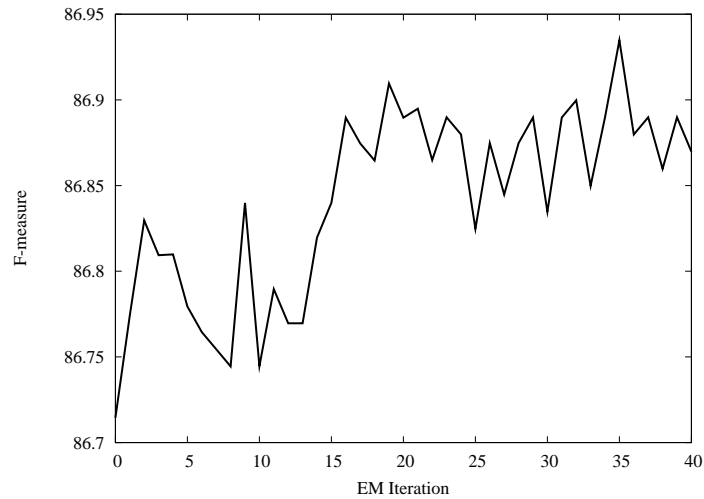


Figure 6.15: F-measure for all sentences of the Penn Treebank development test set with EM models, using §§02–06 as training and a 5% random head-finder for the initial model.

The current answer is that we do not know for certain, but a likely explanation may lie just beneath the surface. When we observe that the standard metrics do not change much over the course of the EM iteration models, we are simply viewing metrics based on the top-scoring parse derivations. It is entirely possible that EM is “reshuffling” the less-likely derivations to produce a model that better explains the observed data. This explanation would not be helpful were it not for the fact that more accurate parses or tree fragments can be found among the top k -scoring derivations for some small $k < 100$. This means that a reestimated model may be a better candidate for parse tree re-ranking experiments, a la (Collins, 2000), than the original model.

Another general observation we may make from the results is that EM reestimation shows a more profound effect when applied to models trained with less data, at least on our development test sets. One likely specific explanation is that EM is compensating for sub-optimal smoothing values chosen in the face of sparse data. A more general version of this explanation is that EM is providing a more *efficient* model—to use that term in both its colloquial and statistical sense—in that it is providing a better-estimated model with the same amount of data.⁷

One final observation is that we are using a training procedure to maximize the likelihood of the observed data, but we have left the decoder as is, meaning that it is still finding the most likely *augmented* tree, $\arg \max_{T, T^+} P(T, T^+, S)$, which is then mapped back down to its unaugmented version for evaluation. Ideally, one would want to marginalize over all possible augmented trees. However, note that this is most likely an intractable problem (we will not present a formal proof here). To begin to see why, observe that this would involve finding the max of a sum: $\arg \max_T \sum_{T^+} P(T, T^+, S)$. There are possible approximations to get around this intractability, one of which will be discussed in Chapter 8.

⁷Actually, I am using the statistical term *efficient* rather informally, in that I am not attempting to prove a mathematical property of the hundreds of thousands of estimated distributions in the model.

Chapter 7

Model analysis: model as data

As we've seen, lexicalized statistical parsing models, such as those built by Black et al. (1992), Magerman (1994), Collins (1999) and Charniak (2000), have been enormously successful, but they also have an enormous complexity. Their success has often been attributed to their sensitivity to individual lexical items, and it is precisely this incorporation of lexical items into features or parameter schemata that gives rise to their complexity. In order to help determine which features are helpful, the somewhat crude-but-effective method has been to compare a model's overall parsing performance with and without a feature. Often, it has seemed that features that are derived from linguistic principles result in higher-performing models (cf. (Collins, 1999)). While this may be true, it is clearly inappropriate to highlight *ex post facto* the linguistically-motivated features and rationalize their inclusion and state how effective they are. A rigorous analysis of features or parameters in relation to the entire model is called for. Accordingly, this chapter aims to provide a thorough analysis of the nature of the parameters in a Collins-style parsing model, with particular focus on the two parameter classes that generate lexicalized modifying nonterminals, for these are where all a sentence's words are generated except for the head word of the entire sentence; also, these two parameter classes have by far the most parameters and suffer the most from sparse data problems. In spite of using a Collins-style model as the basis for analysis, throughout this chapter, we will attempt to present information that is widely applicable because it pertains to properties of the widely-used Treebank (Marcus, Santorini, and Marcinkiewicz, 1993) and lexicalized parsing

models in general.

This chapter also revisits the bilexical dependencies discussed and examined in §6.1. In that section, we not only duplicated Gildea's bigram removal experiment for Collins' Model 2, but we also showed how the decoder used bigram statistics 1.49% of the time on average, thus appearing to confirm the prevailing view that bilexical statistics, while possibly quite powerful, were far too sparse to be very useful for statistical parsing models. However, this chapter will show that such statistics *do* get frequently used for the highest-probability parses, but that when a Collins-style model generates modifier words, the bilexical parameters are so similar to their back-off distributions as to provide almost no extra predictive information.

7.1 Motivation

A parsing model coupled with a decoder is largely an engineering effort. In the end, the performance of the parser with respect to its evaluation criteria—typically accuracy, and perhaps also speed—are all that matter. Consequently, the engineer must understand what the model is doing only to the point that it helps make the model perform better. Given the somewhat crude method of determining a feature's benefit by testing a model with and without the feature, a researcher can argue for the efficacy of that feature without truly understanding its effect on the model. For example, while adding a particular feature may improve parse accuracy, the reason may have little to do with the nature of the feature and everything to do with its canceling other features that were theretofore *hurting* performance. In any case, since this is engineering, the rationalization for a feature is far less important than the model's overall performance increase.

On the other hand, science would demand that, at some point, we analyze the multitude of features in a state-of-the-art lexicalized statistical parsing model. Such analysis is warranted for two reasons: replicability and progress. The first is a basic tenet of most sciences: without proper understanding of what has been done, the relevant experiment(s) cannot be replicated and therefore verified. The second has to do with the idea that, when a discipline matures, it can be difficult to determine what new features can provide the most gain (or *any* gain, for that matter). A thorough

analysis of the various distributions being estimated in a parsing model allows researchers to discover what is being learned most and least well. Understanding what is learned most well can shed light on the *types* of features or dependencies that are most efficacious, pointing the way to new features of that type. Understanding what is learned least well defines the space in which to look for those new features.

In the course of analyzing specific parameters, we will primarily be using two information-theoretic metrics. The first is entropy, which can be interpreted as a measure of a parameter’s closeness to the uniform distribution. Entropy thus provides a measure of a parameter’s efficacy in an absolute sense, for if a distribution is far from uniform, it generally provides more discriminating information to the model compared to a distribution that is closer to uniform: it is “good” when some events have vastly different probabilities than others, because it makes the event space more separable. The second metric is Kullback-Leibler divergence and the related metric of Jensen-Shannon divergence. These divergence metrics provide a kind of distance between two distributions, and are thus useful when comparing one parameter (estimated distribution) to another in the model.

7.2 Frequencies

7.2.1 Definitions and notation

We will refer to any estimated distribution as a *parameter* that has been instantiated from a *parameter class*. For example, in an n -gram language model, $p(w_i | w_{i-1})$ is a parameter class, whereas the estimated distribution $\hat{p}(\cdot | \text{the})$ is a particular parameter from this class, consisting of estimates of every word that can follow the word “the”.

For the analysis described here, we used the baseline model described in 3. As a reminder, our emulation of Collins’ Model 2 has eleven parameter classes, each of which employs up to three back-off levels, where back-off level 0 is just the “un-backed-off” maximal context history. In other words, a smoothed probability estimate is the interpolation of up to three different unsmoothed estimates. The notation and description for each of these parameter classes is shown in Table 7.1.

Notation	Description	No. of back-off levels
P_H	Generates unlexicalized head child given lexicalized parent	3
P_{subcat_L}	Generates subcat bag on left side of head child	3
P_{subcat_R}	Generates subcat bag on right side of head child	3
$P_M (P_{M,NPB})$	Generates partially-lexicalized modifying nonterminal (with NPB parent)	3
$P_{M_w} (P_{M_w,NPB})$	Generates head word of modifying nonterminal (with NPB parent)	3
$P_{prior_{NT}}$	Priors for nonterminal conditioning on its head word and part of speech	2
$P_{prior_{lex}}$	Priors for head word/part of speech pairs (unconditional probabilities)	1 [†]
$P_{TOP_{NT}}$	Generates partially-lexicalized child of +TOP+	1
P_{TOP_w}	Generates the head word for children of +TOP+	2

Table 7.1: All eleven parameter classes in our emulation of Collins’ Model 2. [†]Conditioning context is a static dummy variable used to estimate unconditional probabilities.

7.2.2 Basic frequencies

Before looking at the number of parameters in the model, it is important to bear in mind the amount of data on which the model is trained and on which actual parameters will be induced from parameter classes. The standard training set for English consists of Sections 02–21 of the Penn Treebank, which in turn consist of 39,832 sentences with a total of 950,028 word tokens (not including null elements). There are 44,113 unique words (again, not including null elements), 10,437 of which occur 6 times or more.¹ The trees consist of 904,748 brackets with 28 basic nonterminal labels, to which function tags such as -TMP and indices are added in the data to form 1184 observed nonterminals, not including preterminals. After tree transformations, the model maps these 1184 nonterminals down to just 43. There are 42 unique part of speech tags that serve as preterminals in the trees; the model prunes away three of these (”, “ and .).

Induced from these training data, the model contains 727,930 parameters; thus, there are nearly as many parameters as there are brackets or word tokens. From a history-based grammar perspective, there are 727,930 types of history contexts from which futures are generated. However, 401,447 of these are singletons. The average count for a history context is approximately 35.56, while the average diversity is approximately 1.72. The model contains 1,252,280 unsmoothed

¹We mention this statistic because, as we saw in Chapter 3, Collins’ thesis experiments were performed with an unknown word threshold of 6.

Back-off level	P_M			P_{M_w}		
	\bar{c}	\bar{d}	\bar{n}	\bar{c}	\bar{d}	\bar{n}
0	10.268	1.437	7.145	4.413	1.949	2.264
1	558.047	3.643	153.2	60.19	8.454	7.120
2	1169.6	5.067	230.8	21132.1	370.6	57.02

Table 7.2: Average counts and diversities of histories of the P_M and P_{M_w} parameter classes. \bar{c} and \bar{d} are average history count and diversity, respectively. $\bar{n} = \frac{\bar{c}}{\bar{d}}$ is the average number of transitions from a history context to some future.

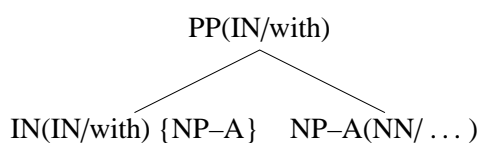


Figure 7.1: A frequent P_{M_w} history context, illustrated as a tree fragment. The ... represents the future that is to be generated given this history.

maximum-likelihood probability estimates ($727,930 \cdot 1.72 \approx 1,252,280$). Even when a given future was not seen with a particular history, it is possible that one of its associated back-off contexts was seen with that future, leading to a non-zero smoothed estimate. The total number of possible non-zero smoothed estimates in the model is 562,596,053. Table 7.2 contains count and diversity statistics for the two parameter classes on which we will focus much of our attention, P_M and P_{M_w} . Note how the maximal-context back-off levels (level 0) for both parameter classes have relatively little training: on average, raw estimates are obtained with history counts of only 10.3 and 4.4 in the P_M and P_{M_w} classes, respectively. Conversely, observe how drastically the average number of transitions \bar{n} increases as we remove dependence on the head word going from back-off level 0 to 1.

7.2.3 Exploratory data analysis: a common distribution

To begin to get a handle on these distributions, particularly the relatively poorly-trained and/or high-entropy distributions of the P_{M_w} class, it is useful to perform some exploratory data analysis. Figure 7.1 illustrates the 25th-most-frequent P_{M_w} history context as a tree fragment. In the top-down model, the following elements have been generated:

- a parent nonterminal PP(IN/with) (a PP headed by the word with with the part-of-speech

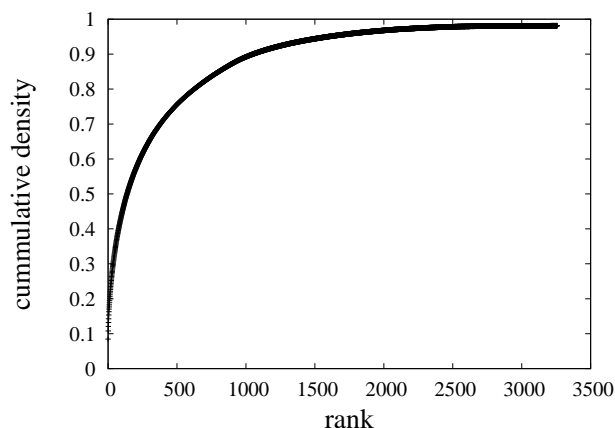


Figure 7.2: Cumulative density function for the P_{M_w} history context illustrated in Figure 7.1.

tag IN)

- the parent’s head child IN
- a right subcat bag containing NP-A (a single NP argument must be generated somewhere on the right side of the head child)
- a partially-lexicalized right-modifying nonterminal

At this point in the process, a P_{M_w} parameter conditioning on all of this context will be used to estimate the probability of the head word of the NP-A(NN), completing the lexicalization of that nonterminal. If a candidate head word was seen in training in this configuration, then it will be generated conditioning on the full context that crucially includes the head word `with`; otherwise, the model will back off to a history context that does not include the head word.

In Figure 7.2, we plot the cumulative density function of this history context. We note that of the 3258 words with non-zero probability in this context, 95% of the probability mass is covered by the 1596 most likely words.

In order to get a better visualization of the probability distribution, we plotted smoothed probability estimates versus the training-data frequencies of the words being generated. Figure 7.3(a) shows smoothed estimates that make use of the full context (*i.e.*, include the head word `with`) wherever possible, and Figure 7.3(b) shows smoothed estimates that do not use the head word. Note how the plot in Figure 7.3(b) appears remarkably similar to the “true” distribution of 7.3(a).

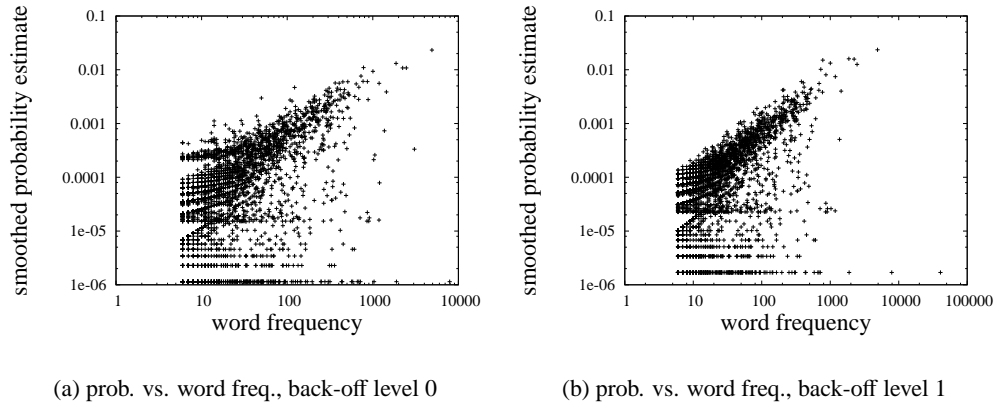


Figure 7.3: Probability versus word frequency for head words of NP-A(NN) in the PP construction.

P_H	0.2516	$P_{TOP_{NT}}$	2.517
P_{subcat_L}	0.02342	P_{TOP_w}	2.853
P_{subcat_R}	0.2147		
P_M	1.121		
P_{M_w}	3.923		

Table 7.3: Average entropies for each parameter class.

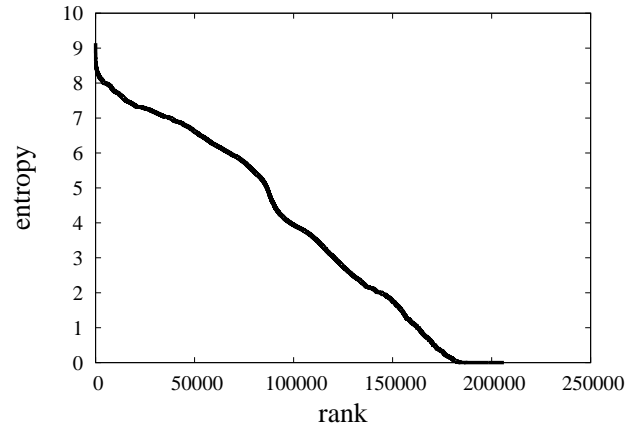
7.3(b) looks like a slightly “compressed” version of 7.3(a) (in the vertical dimension), but the shape of the two distributions appears to be roughly the same. This observation will be confirmed and quantified by the experiments of §7.4.²

7.3 Entropies

A good measure of the discriminative efficacy of a parameter is its entropy. Table 7.3 shows the average entropy of all distributions for each parameter class.³ By far the highest average entropy is for the P_{M_w} parameter class.

²The astute reader will further note that the plots in Figure 7.3 both look bizarrely truncated with respect to low-frequency words. This is simply due to the fact that all words below a fixed frequency are generated as the +UNKNOWN+ word.

³The decoder makes use of two additional parameter classes that jointly estimate the prior probability of a lexicalized nonterminal; however, these two parameter classes are not part of the generative model.

Figure 7.4: Entropy distribution for the P_{M_w} parameters.

Back-off level	P_M				P_{M_w}			
	min	max	avg	median	min	max	avg	median
0	3.080E-10	4.351	1.128	0.931	4.655E-8	9.141	3.904	3.806
1	4.905E-7	4.254	0.910	0.667	2.531E-6	9.120	4.179	4.224
2	8.410E-4	3.501	0.754	0.520	0.002	8.517	3.182	2.451
Overall	3.080E-10	4.351	1.121	0.917	4.655E-8	9.141	3.922	3.849

Table 7.4: Entropy distribution statistics for P_M and P_{M_w} .

Having computed the entropy for every distribution in every parameter class, we can actually plot a “meta-distribution” of entropies for a parameter class, as shown in Figure 7.4. As an example of one of the data points of Figure 7.4, consider the history context explored in the previous section. While it may be one of the most frequent, it also has the highest entropy at 9.141 bits, as shown by Table 7.4. This value not only confirms but quantifies the long-held intuition that PP-attachment requires more than just the local phrasal context; it is, *e.g.*, precisely why the PP-specific features of (Collins, 2000) were likely to be very helpful, as cases such as these are among the most difficult that the model must discriminate. In fact, of the top 50 of the highest-entropy distributions from P_{M_w} , 25 involve the configuration $PP \rightarrow IN(IN/\langle prep \rangle) NP-A(NN/\dots)$, where $\langle prep \rangle$ is some preposition whose tag is IN. Somewhat disturbingly, these are also some of the most frequent constructions.

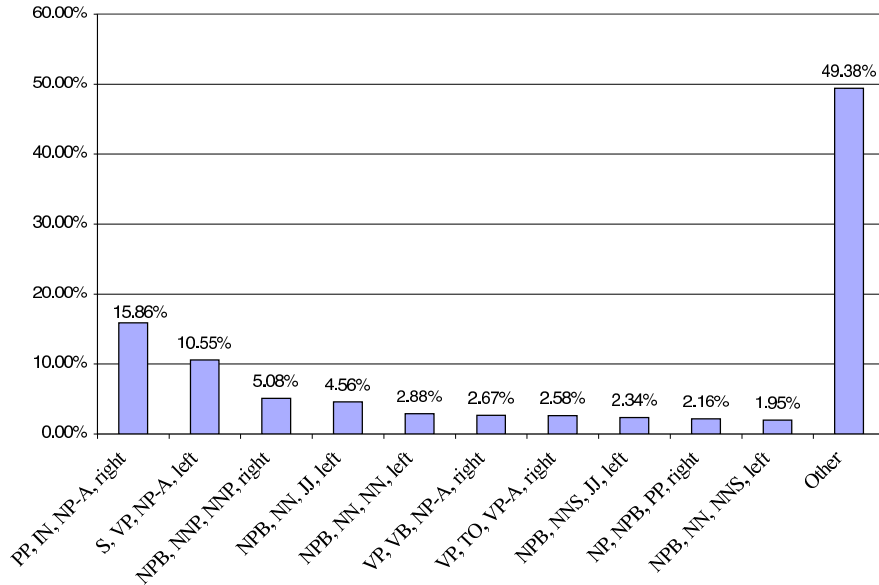


Figure 7.5: Total modifier word-generation entropy broken down by parent, head, modifier triple. The direction word indicates whether the modifier is a left or right modifier.

To gauge roughly the importance of these high-frequency, high-entropy distributions, we performed the following analysis. Assume for the moment that every word-generation decision is roughly independent from all others (this is clearly not true, given head-propagation). We can then compute the total entropy of word-generation decisions for the entire training corpus via

$$H_{P_{M_w}} = \sum_{c \in P_{M_w}} f(c) \cdot H(c) \quad (7.1)$$

where $f(c)$ is the frequency of some history context c and $H(c)$ is that context's entropy. The total modifier word-generation entropy for the corpus with the independence assumption is 3,903,224 bits. Of these, the total entropy for contexts of the form $PP \rightarrow IN \ NP-A$ is 618,640 bits, representing a sizable 15.9% of the total entropy, and the single largest percentage of total entropy of any parent-head-modifier triple (see Figure 7.5).

On the opposite end of the entropy spectrum, there are tens of thousands of P_{M_w} parameters

with extremely low entropies, mostly having to do with extremely low-diversity, low-entropy part-of-speech tags, such as DT, CC, IN or WRB. Perhaps even more interesting is the number of distributions with *identical* entropies: of the 206,234 distributions, there are only 92,065 unique entropy values. Distributions with the same entropy are all candidates for removal from the model, because most of their probability mass resides in the back-off distribution. Many of these distributions are low- or one-count history contexts, justifying the common practice of removing transitions whose history count is below a certain threshold. This practice could be made more rigorous by relying on distributional similarity. Finally, we note that the most numerous low-entropy distributions (that are not trivial) involve generating right-modifier words of the head child of an SBAR parent. The model is able to learn these constructions extremely well, as one might expect.

7.4 Distributional similarity and bilexical statistics

We now return to the issue of bilexical statistics. As alluded to earlier, Gildea (2001) performed an experiment with his partial reimplementation of Collins' Model 1 in which he removed the maximal-context back-off level from P_{M_w} , which effectively removed all bilexical statistics from his model. Gildea observed that this change resulted in only a 0.5% drop in parsing performance. There were two logical possibilities for this behavior: either such statistics were not getting used due to sparse data problems, or they were not informative for some reason. The prevailing view of the NLP community had been that bilexical statistics were sparse, and Gildea (2001) adopted this view to explain his results. Subsequently, we duplicated Gildea's experiment with a complete emulation of Collins' Model 2, and found that when the decoder requested a smoothed estimate involving a bigram when testing on held-out data, it only received an estimate that made use of bilexical statistics a mere 1.49% of the time, as we saw in §6.1. The conclusion at the time of that result was that the minuscule drop in performance from removing bigrams must have been due to the fact that they were barely able to be used. In other words, it appeared that bigram coverage was not nearly good enough for bigrams to have an impact on parsing performance, seemingly confirming the prevailing view.

But the 1.49% figure does not tell the whole story. The parser pursues many incorrect and ultimately low-scoring derivations in its search (in this case, using probabilistic CKY). So rather than asking how many times the decoder makes use of bigram statistics *on average*, a better question is to ask how many times the decoder can use bigram statistics *while pursuing the top-ranked derivation*. To answer this question, we used our parser to *constrain-parse* its own output. That is, having trained it on Sections 02–21, we used it to parse Section 00 of the Penn Treebank (the canonical development test set) and then *re-parse* that section using its own highest-scoring trees (without lexicalization) as constraints, so that it only pursued derivations consistent with those trees. As it happens, the number of times the decoder was able to use bigram statistics shot up to 28.8% overall, with a rate of 22.4% for NPB constituents.

So, bigram statistics *are* getting used; in fact, they are getting used more than 19 times as often when pursuing the highest-scoring derivation as when pursuing any derivation on average. And yet there is no disputing the fact that their use has a surprisingly small effect on parsing performance. The exploratory data analysis of §7.2.3 suggests an explanation for this perplexing behavior: the distributions that include the head word versus those that do not are so similar as to make almost no difference in terms of parse accuracy.

7.4.1 Distributional similarity

A useful metric for measuring distributional similarity, as explored by (Lee, 1999), is the *Jensen-Shannon divergence* (Lin, 1991):

$$JS(p \parallel q) = \frac{1}{2} \left[D(p \parallel \text{avg}_{p,q}) + D(q \parallel \text{avg}_{p,q}) \right] \quad (7.2)$$

where D is the Kullback-Leibler divergence (Cover and Thomas, 1991) and where

$$\text{avg}_{p,q} = \frac{1}{2} (p(A) + q(A))$$

for an event A in the event space of at least one of the two distributions. One interpretation for the Jensen-Shannon divergence due to Slonim, Friedman, and Tishby (2002) is that it is related to the log-likelihood that “the two sample distributions originate by the most likely common source,” relating the quantity to the “two-sample problem”.

In our case, we have $p = p(y | x_1, x_2)$ and $q = p(y | x_1)$, where y is a possible future and x_1, x_2 are elements of a history context, with q representing a back-off distribution using less context. Therefore, whereas the standard *JS* formulation is agnostic with respect to its two distributions, and averages them in part to ensure that the quantity is defined over the entire space, we have the prior knowledge that one history context is a superset of the other, that $\langle x_1 \rangle$ is defined wherever $\langle x_1, x_2 \rangle$ is. In this case, then, we have a definition for the Jensen-Shannon divergence that is generalized to the multiple distributions that include an extra history component:

$$\begin{aligned} JS(p \parallel q) &= \sum_{x_2} p(x_2) \cdot D(p(y | x_1, x_2) \parallel p(y | x_1)) \\ &= E_{x_2} D(p(y | x_1, x_2) \parallel p(y | x_1)) \end{aligned} \quad (7.3)$$

An interpretation in our case is that this is the expected number of bits x_2 gives you when trying to predict y .⁴ If we allow x_2 to represent an arbitrary amount of context, then the Jensen-Shannon divergence $JS_{b \leftarrow a} = JS(p_b \parallel p_a)$ can be computed for any two back-off levels, where a, b are back-off levels s.t. $b < a$ (meaning p_b is a distribution using more context than p_a). The actual value in bits of the Jensen-Shannon divergence between two distributions should be considered in relation to the number of bits of entropy of the more detailed distribution; that is, $JS_{b \leftarrow a}$ should be considered relative to $H(p_b)$. Having explored entropy in §7.3, we will now look at some summary statistics for *JS* divergence.

7.4.2 Results

We computed the quantity in Equation 7.3 for every parameter in P_{M_w} that used maximal context (contained a head word) and its associated parameter that did not contain the head word. The results are listed in Table 7.5. Note that, for this parameter class with a median entropy of 3.8 bits, we have a median *JS* divergence of only 0.097 bits. The distributions are so similar that the 28.8% of the time that the decoder uses an estimate based on a bigram, it might as well be using one that does not include the head word.

⁴Or, following from Slonim et al.’s interpretation, this quantity is the (negative of the) log-likelihood that all distributions that include an x_2 component come from a “common source” that does *not* include this component.

	min	max	avg.	median
$JS_{0 \leftarrow 1}$	2.729E-7	2.168	0.1148	0.09672
$JS_{1 \leftarrow 2}$	0.001318	1.962	0.6929	0.6986
$JS_{0 \leftarrow 2}$	0.001182	1.180	0.3774	0.3863

Table 7.5: Jensen-Shannon statistics for back-off parameters in P_{M_w} .

7.5 Distributional Similarity and Parameter Selection

The analysis of the previous two sections provides a window onto what types of parameters the parsing model is learning most and least well, and onto what parameters carry more and less useful information. Having such a window holds the promise of discovering new parameter types or features that would lead to greater parsing accuracy; such is the scientific, or at least, the forward-minded research perspective.

7.5.1 Metric and experiments

From a much more purely engineering perspective, one can also use the analysis of the previous two sections to identify individual parameters that carry little to no useful information and simply remove them from the model. Specifically, if p_b is a particular distribution and p_{b+1} is its corresponding back-off distribution, then one can remove all parameters p_b such that

$$\frac{JS(p_b \| p_{b+1})}{H(p_b)} < t, \quad (7.4)$$

where $0 < t < 1$ is some threshold.

One might ask why the JS divergence is divided by the entropy of the p_b distribution. The answer stems from the nature of KL divergence, of which JS divergence is a generalization. One interpretation of KL divergence is the average number of “wasted” bits coding samples over a channel drawn from p_b by using p_{b+1} . The intuition is that we want to divide this number by the expected number of bits needed to encode p_b . More formally, we would like to know if the relative reduction in perplexity from p_{b+1} to p_b is less than some fixed, fractional exponentiation of the perplexity of the “true” model p_b . If it is, then we eliminate p_b . To see this in terms of KL

divergence for some event space \mathcal{X} , observe that

$$\begin{aligned} D(p_b \| p_{b+1}) &= \sum_{x \in \mathcal{X}} p_b(x) \lg \frac{p_b(x)}{p_{b+1}(x)} \\ &= - \sum_{x \in \mathcal{X}} p_b(x) [\lg p_{b+1}(x) - \lg p_b(x)] \end{aligned} \quad (7.5)$$

so

$$2^{D(p_b \| p_{b+1})} = \frac{2^{-\sum_x p_b(x) \lg p_{b+1}(x)}}{2^{-\sum_x p_b(x) \lg p_b(x)}}. \quad (7.6)$$

Assuming that p_b is the “true” distribution because it uses more context, then (7.6) is the perplexity of the model under p_{b+1} relative to the true perplexity. Our metric using KL divergence in place of JS divergence is thus

$$\frac{D(p_b \| p_{b+1})}{H(p_b)} < t \quad (7.7)$$

which is the same as

$$\begin{aligned} 2^{D(p_b \| p_{b+1})} &< 2^{t H(p_b)} \\ \frac{2^{-\sum_x p_b(x) \lg p_{b+1}(x)}}{2^{-\sum_x p_b(x) \lg p_b(x)}} &< \left(2^{H(p_b)}\right)^t, \end{aligned} \quad (7.8)$$

where $0 \leq t \leq 1$.

Table 7.6 shows the results of pruning parameters from our baseline model using the metric in Equation 7.4 with a threshold of 0.06. Not all parameters that fell below this threshold were pruned, as we were somewhat conservative in the pruning strategy: if a parameter p_a was not pruned, then no attempt was made to prune any of the other parameters in its back-off chain (p_b , where $b > a$). Backing off to more general histories means that the parameters of the model form a tree, and so this strategy is like pruning a physical tree: you do not try to prune the inner branches without having pruned the outer ones.⁵ To our knowledge, this is the first example of detailed parameter selection in the context of a generative lexicalized statistical parsing model. The consequence is a significantly smaller model that performs *with no loss of accuracy* compared to the full model.⁶

⁵Also, pruning by distributional similarity to a back-off distribution means that parameters at the last level of the back-off chain are never candidates for pruning by this strategy. This is another way that our strategy is conservative, in a sense.

⁶None of the differences between the Model 2-emulation results and the reduced model results is statistically significant. The threshold 0.06 was arrived at via experimentation, and was the largest threshold ± 0.005 such that parsing accuracy did not suffer.

≤ 40 words				
Model	§00		§23	
	LR	LP	LR	LP
m3	n/a	n/a	88.6	88.7
m2-emu	89.9	90.0	88.8	88.9
reduced	90.0	90.2	88.7	88.9
all sentences				
Model	§00		§23	
	LR	LP	LR	LP
m3	n/a	n/a	88.0	88.3
m2-emu	88.8	89.0	88.2	88.3
reduced	89.0	89.0	88.0	88.2

Table 7.6: Parsing results on Sections 00 and 23 with Collins’ Model 3, our emulation of Collins’ Model 2 and the reduced version at a threshold of 0.06. LR = labeled recall, LP = labeled precision.⁶

Reducing the size of the model also naturally reduces the memory footprint of the parser, which in turn increases spatial locality and therefore increases speed. In fact, the decoder with the reduced model parses §00 1.7–1.8 times as quickly as the full model. While some of this speedup is due to the reduced memory footprint, the vast majority of it is due to the fact that the parameter removal leads to a narrowing of the probability densities (as shown graphically by the “compressed” vertical dimension in Figure 7.3(b)), which allows the use of a narrower beam. Specifically, the above results may be obtained with a beam of $10^{3.5}$ instead of the default beam of 10^4 .⁷

Further insight is gained by looking at the percentage of parameters removed from each parameter class. The results of Chapter 6 suggested that the power of Collins-style parsing models did not lie primarily with the use of bilexical dependencies as was once thought, but in *lexico-structural* dependencies, that is, predicting syntactic structures conditioning on head words. The percentages of Table 7.7 provide even more concrete evidence of this assertion, for whereas nearly a third of the P_{M_w} parameters were removed, a much smaller fraction of parameters were removed from the P_{subcat_L} , P_{subcat_R} and P_M classes that generate structure conditioning on head words. Note also that removal of a P_{M_w} parameter results in removal of far more “information” from the model than, say,

⁷Two of the 1917 sentences of §00 of the Penn Treebank cannot be parsed with a beam of $10^{3.5}$. In the case of these failures, the parser tries a wider beam of 10^4 , which, of course succeeds. The number of iterations and the increment for beam widening are user-selectable run-time parameters in the parsing engine. Modulo these two sentences, parsing §00 with a beam of 10^4 yields virtually identical results to parsing with $10^{3.5}$.

Param. class	removed	total	% removed
P_H	11081	81964	13.5%
P_{subcat_L}	529	79139 [†]	0.67%
P_{subcat_R}	1408	79139 [†]	1.8%
P_{TOP_w}	12	52705	0.023%
P_M	27068	267835	10.1%
P_{M_w}	61981	211023	29.4%
Total	102079	692666 [†]	14.7%

Table 7.7: Percentage of parameters removed from each parameter class for the 0.06-reduced model. [†]The subcat contexts were counted once.

a P_{TOP_w} that uses much less context.

7.5.2 Related work

The use of information-theoretic metrics for parameter weighting or parameter selection is nothing new. Quinlan (1993) proposed using an *information gain* (IG) parameter weighting metric for class-labeling problems:

$$w_i = H(C) - \sum_{v \in V_i} p(v) \cdot H(C|v), \quad (7.9)$$

where w_i is the weight associated with the i th feature function, V_i is the set of possible values of the feature, and $H(C)$ is the entropy of the distribution of classes. The idea is to compute the reduction in entropy due to the presence of the feature; (7.9) is also more commonly called mutual information, denoted $I(C; V_i)$. Quinlan further observed that (7.9) on its own would be biased toward feature functions with high numbers of values, since they would inherently have a higher maximum possible entropy. The proposed solution was to divide by the entropy of the feature values, resulting in a *gain ratio*:

$$w_i = \frac{H(C) - \sum_{v \in V_i} p(v) \cdot H(C|v)}{-\sum_{v \in V_i} p(v) \lg(v)}. \quad (7.10)$$

This metric was also incorporated into the Tilburg memory-based learning system, TiMBL (Daelemans, Zabrel, and van der Sloot, 2002).

More directly relevant to the parameter selection metric we have employed here, Stolcke (1998) employed an entropy-based metric for parameter selection in an n -gram language model, resulting

in a 26% reduction of the number of parameters with no increase in speech recognition error. In that work, each n -gram in the model was considered independently as to how it affected the overall perplexity of the entire model. If pruning a parameter led to an increase in relative perplexity⁸ that was less than some fixed threshold θ , then the parameter was removed. After all such parameters were removed, smoothing weights were recomputed.

Stolcke’s metric is similar to ours in that it relies on relative entropy, but is crucially different in that the perplexity of the entire model was recomputed for each parameter. Because of the difficulty of doing this for a parsing model that does not have the relative simplicity of a Markov model, we have a metric solely based on distributional similarity.

7.6 Discussion: a problem related to label bias

Conditional models can theoretically suffer from a “label bias” problem, which occurs from a lack of sensitivity to the input sequence. In the extreme case, suppose the decoder for a conditional model reaches a state that has only one outgoing transition. Now further suppose that the sole available next state is very unlikely given the current input symbol. In spite of this, the model will assign high (1.0) probability to the one available transition, because it is the only choice available and since the model by construction does not condition on the input. This problem in the context of NLP models has been the subject of recent discussion (Lafferty, Pereira, and McCallum, 2001; Klein and Manning, 2002a). A generative model cannot suffer from this problem because it explicitly models the input given the hidden state, meaning it is able to multiply a low probability for state sequences that are not supported by the current input symbol.

Examining the lower-entropy P_{M_w} distributions revealed that, in many cases, the model was not so much learning how to disambiguate a given syntactic/lexical choice, but simply not having much to learn. For example, once a partially-lexicalized nonterminal has been generated whose tag is fairly specialized, such as IN, then the model has “painted itself into a lexical corner”, as it were (the extreme example is TO, a tag that can only be assigned to the word to). So, suppose the model generates some tag that is quite likely given the current history of decisions and that is most

⁸Relative perplexity is the exponentiation of relative entropy, or Kullback-Leibler divergence.

often observed with the word about to be generated. Further suppose that the word is quite *unlikely* in the context of the current history, but has never been observed in that history. The model will then simply use $\hat{p}(w|t)$, which is high. In other words, the model is unable to *discount* a bad theory due to its lack of sensitivity to the input in the face of the rich history of decisions. This might be described as a “lack of observation bias” problem.

Of course, just because there is a “lack of observation bias” does not necessarily mean there is a problem. If the decoder pursues a derivation to a nonterminal/part-of-speech tag preterminal that has an extremely low entropy distribution for possible head words, then there is certainly a chance that it will get “stuck” in a potentially bad derivation. This is of particular concern when a head word—which the top-down model generates at its highest point in the tree—influences an attachment decision. However, inspecting the low-entropy word-generation histories of P_{M_w} revealed that almost all such cases are when the model is generating a preterminal, and are thus of little to no consequence vis-a-vis syntactic disambiguation.

7.7 Chinese model analysis

The above analysis on the English model can just as easily be performed on the Chinese model. This section presents an analysis of the Chinese model, and does a comparison with the English analysis where appropriate.

7.7.1 Entropy analysis

In English, in Figure 7.5, we saw how the construction $PP \rightarrow IN NP-A$ was not only among the most frequent in the corpus, but also the most entropic, constituting 15.9% of the word-generation entropy of the training corpus. The Chinese corpus as viewed through the lens of the model exhibits a much different distribution, as shown in Figure 7.6. The analogous construction ranks only sixth with the Chinese model and corpus.

Looking at the constructions that contribute most to the word-generation entropy, we find $NPB, NN, NN, left$. This is both to be expected and to be ignored: common nouns constitute the

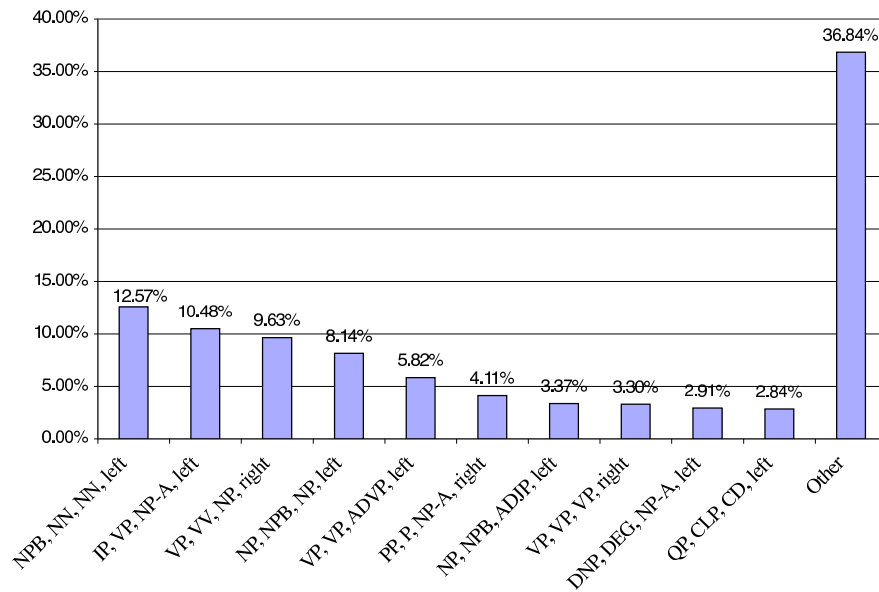


Figure 7.6: Total modifier word-generation entropy for the CTB3.0 training corpus, broken down by parent,head,modifier triple. The direction word indicates whether the modifier is a left or right modifier.

largest set of vocabulary items and occur most frequently, hence the expectation of high word-generation entropy. This construction is to be ignored because it is part of the special Markov model for generating base NPs, and therefore *not* part of entropy related to the difficulty of syntactic disambiguation.

The construction of rank 2 is *IP, VP, NP-A, left*, which essentially is the situation of generating a noun subject for a verb. A manual inspection of the actual verbs of these constructions (reverse-sorted by entropy, of course) revealed a large number that were light verbs or other verbs carrying very little semantic information (see Figure 7.7 for a list of these verbs and their glosses).⁹ In English, the analogous construction of *S, VP, NP-A, left* is of rank 2 (see Figure 7.5), and manually inspecting the verbs of these constructions revealed that the few that were as entropic as the PP constructions were auxiliaries like “can” and “will”. As one would expect from the “model as data” approach, these data say as much about the differences between the English and Chinese training corpora as they do about their respective models. The head-finding heuristics for English search the children of a VP from left-to-right, picking individual verbs (such as anything tagged with VBZ or MD) as the head if possible; if there is no such preterminal to choose, then the next rule says to pick the leftmost VP child. By contrast, the analogous rule used for the Chinese model does the opposite, first searching left-to-right for a VP, and only if that search fails, searching for the various verb tags as the head. The different heuristics may account for some of the observed difference in ranking of constructions shown in Figures 7.5 and 7.6. Beyond this difference, it appears to be the case that in both models, the light verbs are less felicitous for reducing entropy than those with more semantic content. This does not necessarily mean, however, that the head rules should be modified, for while a modification may yield lower word-generation entropy in some cases, it may also increase it in others, as well as increase sparse data problems. One advantage to using auxiliaries as heads, at least in English, is that they allow for number agreement via the “bi-tag” dependencies in the model, and, as closed-class items, sparse data is not much of an issue.

We are, of course, ultimately interested in parsing accuracy. So, we trained and tested our Chinese parser on the CTB3.0 data using the head rules specified in (Sun and Jurafsky, 2004), which have a heuristic for VP parents that is much more similar to the English heuristic, in that it

⁹Thanks to Bert Xue for examining these verbs and providing glosses.

包括	include
进行	conduct
会	will
作为	as/be
给予	give
能	can
可	may
减少	decrease
开始	begin
维持	maintain
确定	determine
强化	strengthen
进行	conduct
成为	become
没有	not have
作	as/be
拥有	*own
到	reach
购买	*purchase
使	cause
可以	may
恢复	resume
建设	*construct
为	as/be
出口	*export
要	will
属	belong
获得	get
建成	*build into
可能	may
建	build
前往	go
受到	receive
取消	cancel

Figure 7.7: Chinese verbs associated with high-entropy P_{M_w} parameters, in decreasing order of the entropy of their respective parameters. The five asterisked verbs are the only verbs in this set that are unambiguously “heavy”.

Head rules used	≤40 words					F
	LR	LP	\overline{CB}	0 CB	≤2CB	
default	79.6	82.9	1.36	50.8	80.1	81.2
sun	79.6	82.8	1.37	50.6	80.0	81.2
	≤100 words					
default	77.5	80.6	2.10	42.5	70.2	79.0
sun	77.7	80.8	2.06	42.5	69.6	79.2

Table 7.8: Comparison of two different head-finding rule sets on CTB3.0. “sun” refers to the head rules proposed in Sun and Jurafsky (2004). None of the differences in LR or LP is statistically significant.

searches for a VP child only after first doing searches for the various verb tag preterminals.¹⁰ The parsing results were nearly identical to the results with our default rule set, as shown in Table 7.8. This was expected, both because of the reasons mentioned above and because, as we saw back in §6.1.1, head choice is not terribly critical to parsing accuracy.¹¹

On the other end of the entropy spectrum, Chinese exhibited a pattern similar to English with respect to the least entropic parameters. They tended to be associated with low-entropy part-of-speech tags, as well as right modifiers of the head child of a CP parent.

7.7.2 Parameter selection

The parameter selection criterion of Equation 7.4 was applied to Chinese. As with English, we experimented with several threshold values, and possibly by coincidence 0.06 was optimal. Table 7.9 presents the results of the full CTB3.0-trained model and the 0.06-reduced model. Also as with the reduced English model, there was no loss in parsing accuracy.

¹⁰Actually, the head rules specified in (Sun and Jurafsky, 2004) are not exhaustive, and so for each parent for which a heuristic was not provided, we used the heuristic from our default rule set.

¹¹It is unfortunate but necessary to mention that the parsing results reported in (Sun and Jurafsky, 2004) should be ignored. The paper has two flaws, one major, one minor. The major flaw is that the parsing experiments were performed using gold-standard part-of-speech tags as input along with the words, even though the accuracy numbers were compared in the paper to results that had made use of automatic part-of-speech taggers, such as `mxpost`. The minor flaw was that the authors claimed to have the first results of a model based on the Collins model, even though results using this core model had been presented at the same conference two years earlier (Bikel, 2002).

Model	≤ 40 words					
	LR	LP	\overline{CB}	0 CB	$\leq 2CB$	F
default	79.6	82.9	1.36	50.8	80.1	81.2
reduced	79.5	83.0	1.36	51.3	80.0	81.2
≤ 100 words						
default	77.5	80.6	2.10	42.5	70.2	79.0
reduced	77.3	80.7	2.10	42.8	70.3	79.0

Table 7.9: CTB3.0 parsing results with the full model and the 0.06-reduced model.

Param. class	English			Chinese		
	removed	total	% removed	removed	total	% removed
P_H	11081	81964	13.5%	3660	40683	9.00%
P_{subcat_L}	529	79139 [†]	0.67%	166	41249 [†]	0.40%
P_{subcat_R}	1408	79139 [†]	1.8%	110	41249 [†]	0.27%
P_{TOP_w}	12	52705	0.023%	1	22315	0.004%
P_M	27068	267835	10.1%	12591	122943	10.2%
P_{M_w}	61981	211023	29.4%	13561	68237	19.9%
Total	102079	692666 [†]	14.7%	30089	295427	10.2%

Table 7.10: Percentage of pruned parameters per class for both the Chinese and English models.
[†]The subcat contexts were counted once.

7.8 A closer look at the pruned parameters

In §7.5, Table 7.7 presented the percentage of parameters pruned from each of the parameter classes of the English model. Table 7.10 gives a side-by-side comparison of those percentages for both the English and Chinese models. Once again, not all parameters are created equal: a P_{M_w} parameter, for example, contains far more information than, say, a P_H parameter, so the overall percentage of parameter reduction does not indicate the true reduction in model complexity. But we may go further than this macro-analysis of the pruned parameters. We would like to know, for example, if there are generalizations or characterizations of the pruned parameters within a particular class. As in much of our analysis work, our focus will be on the P_{M_w} class, since that class has the largest

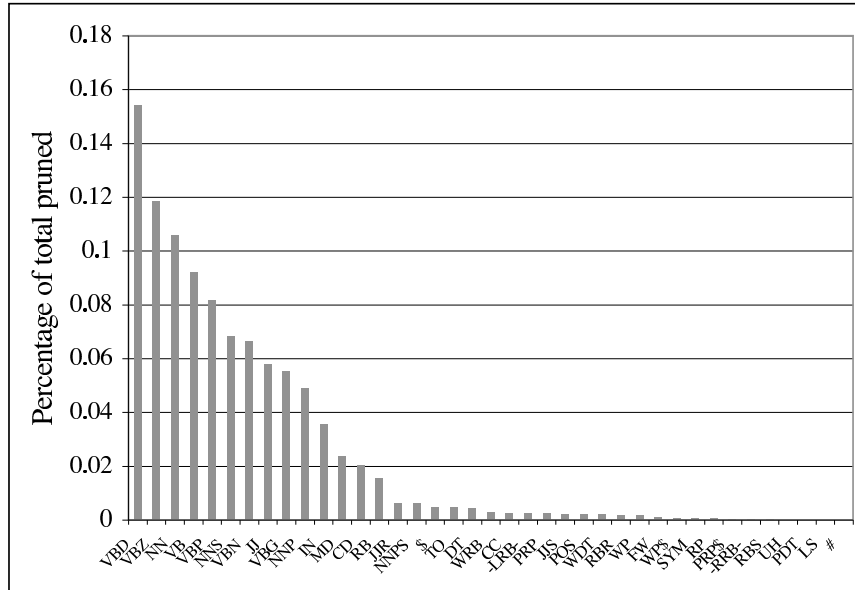


Figure 7.8: Percentage of total number of pruned P_{M_w} parameters, broken down by tag of head word, t_h .

proportion of pruned parameters.¹²

Recall that the only difference between back-off level 0 and 1 of the P_{M_w} parameters is the inclusion/omission of the head child’s head word, w_h . To get an idea of what sorts of words have been pruned (and therefore to get an idea of what sorts of words have proven to be less useful than others), we subdivide the P_{M_w} parameter class into subclasses P_{M_w, t_h} , one for each part of speech of the head word, t_h (“head tag”). Figure 7.8 presents the percentage of total pruned parameters of P_{M_w} broken down by head tag. Of course, the verb and noun categories, being the most frequent, rank the highest of the total percentage of pruned parameters, although it is noteworthy that VBG and VBZ rank higher than the common noun categories NN and NNS, even though such nouns are more frequent. There are clearly larger numbers of progressive aspect (VBG) and present tense (VBZ) verbs where the identity of the word is no more informative (or not much

¹²This is now in the realm of “meta-analysis”, in that the pruning process was itself the result of distributional analysis.

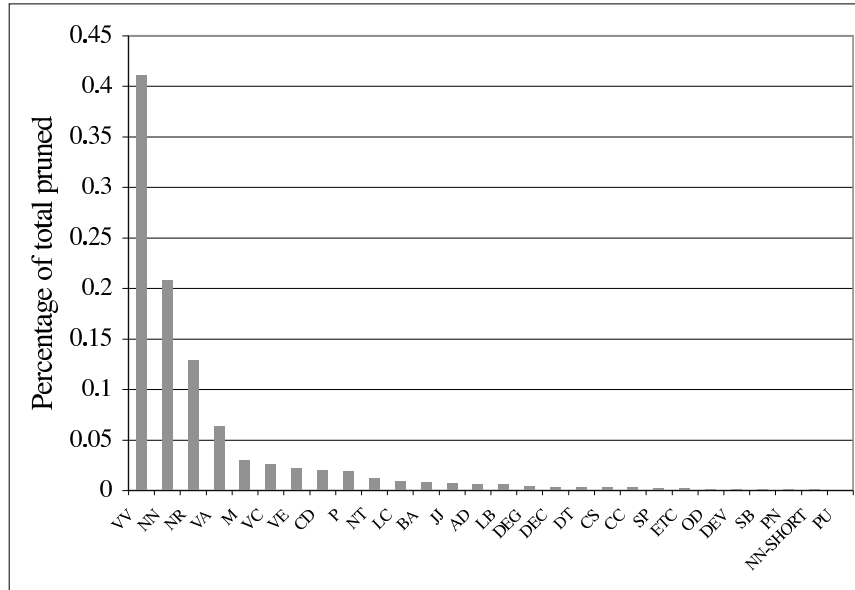


Figure 7.10: Percentage of total number of pruned P_{M_w} parameters in the CTB3.0-trained Chinese model, broken down by tag of head word, t_h .

(VV) and the common noun category (NN) is even more striking than with English. Figure 7.11 shows the percentage of parameters pruned from each P_{M_w, t_h} (akin to Figure 7.9). Just as with English, we notice that the parameter subclasses associated with the more specialized tags have a higher percentage of pruned parameters than the subclasses associated with the general, open-class tags, such as NN and VV.

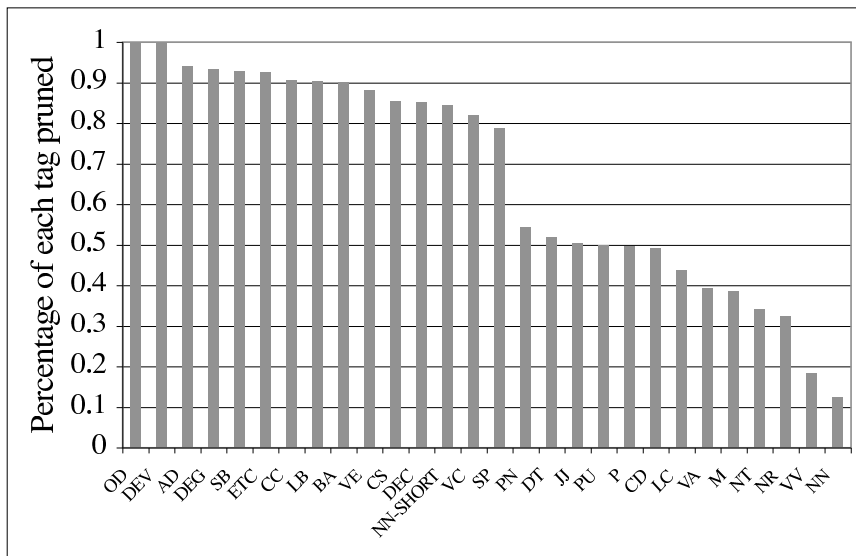


Figure 7.11: Percentage of parameters pruned from each subclass P_{M_w, t_h} .

Chapter 8

Conclusion

8.1 Summary

Parsing models can be hideously complex. We saw some of this complexity in Chapter 3. However, with careful attention, the particular lexicalized parsing model we took as our baseline proved to be separable into a language-independent core and a periphery, and this decomposition was implemented in the software framework described in Chapter 4. Moreover, the decomposition and this framework were exploited to good benefit in multiple languages, as we saw in Chapter 5, particularly in Chinese, for which there is now a large amount of high-quality data.

And yet, the proliferation of parsing model features, even if they are based in linguistics, can lead to unforeseen complexities and interactions. In Chapter 6 we began to analyze the model in the meta-parameter space, teasing apart the effect of certain dependencies. In particular, we began to see the extent to which our baseline model made use of the lexicalization that was long thought to be its greatest source of discriminative power. Since lexicalization is performed initially via heuristics, that chapter also introduced and explored a new EM method to reestimate the model in the augmented tree space.

In Chapter 7, we moved to an even greater level of detail. Recognizing that the model is composed of hundreds of thousands of estimated distributions or parameters, we treated the *model as data*, and accordingly performed exploratory data analysis. The purpose of EDA is to discover

relationships among distributions or other properties of data, relying initially on various visualizations of those data by which the human eye may detect patterns. Indeed, one of these visualizations suggested a strong similarity between the maximal-context word-generation parameters and their back-off counterparts. By explicitly testing for this similarity using an information-theoretic distributional similarity metric, this intuition was both confirmed and quantified. Furthermore, the similarity was employed in a simple metric that allows, for the first time, parameter selection in a generative statistical parsing model, such that 14.7% of the total parameters (including 29.7% of the modifier word-generation parameters) were removed with no loss in accuracy.

But the analyst's job is never done, it seems, for once we prune tens of thousands of parameters, they themselves may be treated as data to be analyzed and/or categorized. By doing just that, we were able to see which syntactic categories of words were more useful than others for the parser.

8.2 Interpretability

When statistical models first started gaining ground as tools for NLP problems, some “traditional” NLP researchers derided them for their lack of interpretability. The claim was that rule-based systems could be directly inspected and understood, but that statistical models could not. This was oftentimes a false claim, for while the rules themselves might have been clear, the hand-coded mechanisms to control their interactions often were not. Statistical models seemed to provide the necessary relief from worrying about the interactions of various features, for probability theory provides an excellent mechanism by which to combine information. The hope was that researchers could specify whatever features seemed even remotely plausible, and that, after easily finding an appropriate model, a training procedure would determine their weights and an efficient decoder would search the model for each test instance.

This rosy picture has also not quite come into being, although great strides have been made in recent years incorporating ever-more sophisticated statistical modeling techniques into NLP research. But the interpretability issue has not disappeared; if possible, it has become even more important, given the new-found freedom in declaring features. With several modeling techniques, such as log-linear conditional models, part of the task of interpreting the model is accomplished by

the training procedure, which automatically determines feature weights and thus provides a guide to each feature's "importance" in the overall model. For various reasons, the early history of statistical NLP has been dominated by generative models, which are very convenient probabilistically, but have relatively severe limitations in the types of features that may be encoded. History-based parsing models fall into this class. In spite of the limitations on features, models such as Collins' and Charniak's saw a proliferation in types of dependencies employed.

In this thesis, essentially by treating the model as data, we have begun to untangle the complexity of the dominant class of statistical parsing model. At the highest level, we have decomposed the model into its language-independent core and the language-dependent periphery. We have then investigated the meta-parameter space, developing a way to use EM to reestimate the model in a certain part of this space, and discovering the crucial role played by smoothing/back-off in these models. Finally, in the most detailed analysis, we have explored the efficacy of individual parameters, both in an absolute sense (entropy) and in a relative sense (divergence from other, related distributions in the model). This detailed analysis has allowed for the first time a way to do parameter selection in a generative lexicalized statistical parsing model, such that a significantly smaller model can parse with no loss in accuracy compared to the full model, and at much greater speed. Thus, we now have a clearer picture of how the dominant class of state-of-the-art parsing models works, and we have developed a methodology for analysis of other model types.

8.3 Future work

There are several areas of future work either suggested by or directly building on the work of this thesis. For example, with each new non-English treebank, there is an opportunity to create a language package for the parsing engine described in Chapter 4 and tested in Chapter 5. With each new language, it would make sense to test new dependencies, which is also straightforward with the engine. Such work is, in fact, already under way at several institutions, including the University of Pennsylvania.

Cautionary note Throughout this dissertation, I have intentionally kept speculation to a minimum. In the NLP community, there have been too many instances where speculation in one bit of research has gone untested and has become assumption for other, subsequent research. In fact, part of this thesis has been the refutation of some long-held assumptions. Since this section is devoted to future work, what follows contains much speculation. The reader is strongly urged to treat this speculation with a healthy dose of suspicion, and to not believe any assertion until it has been satisfactorily tested.

8.3.1 Word classes

The EM method developed in Chapter 6 treated node augmentations as hidden data, but another natural extension of this work would be to treat semantic word classes as hidden. There are several pitfalls that should be avoided with this approach, and with using semantic word classes in general.

My own initial experiments using WordNet classes seemed promising, but failed to improve parsing performance (Bikel, 2000). In those experiments, I merged the intersection of the syntactically-annotated Brown corpus with the SEMCOR corpus, yielding about 230,000 words of data annotated for both syntax and lexical semantics. In the model I developed, a synset was generated for every noun, verb, adjective and adverb in the test data, crucially using both the noun and verb hypernym hierarchy for back-off. In fact, since back-off is defined recursively (see Equation 3.15), I essentially had the back-off hierarchy mirror the hypernym hierarchy. This slavishness to the WordNet-defined hierarchy of senses was a very likely reason parsing did not improve over the baseline model in those experiments. Another likely reason was the inability of the modified Witten-Bell smoothing method to deal well with arbitrary numbers of back-off levels.

More recently, I have done numerous experiments along the lines described above, where semantic word classes were treated as hidden data and a model was reestimated using EM. In these experiments, I used the same Brown/SEMCOR data to train an initial model, but instead of incorporating hypernyms directly into the back-off chain, for each synset-annotated noun and verb, the trainer of the initial model hallucinates a fractional occurrence of each possible hypernym, essentially putting a uniform distribution over possible hypernyms while preserving the marginal

probability of each word. In this way, the hidden data may be thought of as parameters that choose an appropriate “leaf” synset for each noun and verb (adjectives and adverbs were stripped of any synset annotations in preprocessing), and parameters that choose a hypernym from the complete hypernym set of that leaf. The idea is then that EM should provide good estimates for these hidden parameters, specifically finding the right level of lexical “generality” for each syntactic context (by picking the most felicitous hypernym).¹

These WordNet EM experiments proved unfruitful. In addition to the reasons described at the end of Chapter 6, there is now good reason to suspect that WordNet synsets and the hypernym hierarchy are so fine-grained as to not necessarily be suitable either for word sense disambiguation or for syntactic disambiguation, as discussed in great detail by Palmer, Dang, and Fellbaum (2004). In particular, the hypernym hierarchy, rather than coalescing similar senses—as one might expect word clusters to do—oftentimes only serves to reinforce the differences. Palmer *et al.* point out a good example of this problem with the verb *play*, which in WordNet v1.7 has the three “leaf” senses that are all related to playing music, but with three *disjoint* hypernym sets. This situation is not exceptional, but actually rather common. Palmer, Dang, and Fellbaum (2004) provide groupings of WordNet senses that take syntactic alternations and other syntactic criteria into account. The tables at the end of that paper provide further evidence of the disjoint nature of the hypernym hierarchy by showing how syntactically-based groupings of senses often map onto multiple, disjoint hypernym chains. Palmer *et al.* also show how there is significantly higher inter-annotator agreement when measured with these groupings. In all, then, I suspect that such syntactic/semantic groupings are much more likely to be fruitful for incorporation into a parsing model (whose job, after all, is to perform syntactic disambiguation) than the original WordNet senses and their hypernyms. As it happens, new sources of data making use of such syntactico-semantic word classes are now becoming available, such as the PropBank (Kingsbury and Palmer, 2002), and the nascent OntoBank, which is currently in the planning stages.

Even though I believe the use of such data holds great promise, caution is still advised. One of the primary motivations for the incorporation of word clusters into a parsing model was to

¹The model does not contain such hidden parameters explicitly. It is still a top-down, generative model, so there are synset-generation parameters and the P_{M_w} parameters are modified to include dependence on a previously-generated synset for the word.

overcome the sparsity of lexical statistics, especially bilexical statistics. As this thesis has shown, such statistics are used far more often than originally thought, thereby casting doubt on whether the addition of semantic word classes would help in a substantial way. The other primary motivation for using semantic word classes was that they would provide more syntactic sensitivity than our current word classes, which are the Treebank's part-of-speech tags. Before proceeding headlong, then, into the use of such word classes, it would be helpful to do the sort of analysis described in this thesis. If the assumption is that word classes would occasionally override the syntactic preference of a word's part-of-speech tag, then this assumption may be rigorously tested by analyzing the divergence of distributions including the semantic class from those that only include the part-of-speech tag.

There is also no reason to suspect that word classes should simply be “grafted” onto an existing model's dependencies. If we are already getting good use of bigrams mediated by syntactic context, then perhaps we should look toward syntactic trigrams or even tetragrams. Strictly lexical syntactic n -grams have already been shown to be useful for such thorny parsing issues as PP-attachment and coordinated constructions (Collins, 2000). I suspect that “long-range” features that incorporate the selectional restrictions of verbs along with the semantic type of arguments—even if those arguments are complements of, say, prepositions—will not only improve syntactic disambiguation, but will also form a crucial part of the ultimate goal, which is to extract a semantic representation from a sentence.

The foregoing raises an interesting point: with the exception of work in DOP (Bod, 1992), very few of the top-performing parsing models that can take advantage of “global” features truly make use of this property. The feature sets described in (Collins, 2000) and (Charniak, 2000) all have a fairly limited domain of locality, generally not exceeding a grandparent-grandchild dependence. It would therefore be useful to further explore this restricted space of features for the incorporation of word classes. Such an exploration need not even occur within a conditional modeling framework, given the restricted domain of locality. It would be entirely feasible to continue in a history-based framework simply by more tightly coupling the decoder with the model: when longer range features are used, the decoder could shift into k -best parsing mode (or even simply keep *all* chart items associated with longer range features). For example, if there is a feature that involves a dependency between a verb, its object, a preposition and the object of the preposition, the decoder could keep all

PP chart items within the pruning threshold (*i.e.*, not coalesce equivalent items as with the normal dynamic programming algorithm) to allow use of the long-range features later in the decoding process, when constructing VP chart items that cover the span of the PP items. One might ask, why even bother to stick with history-based modeling? Indeed, the great boon of many conditional modeling frameworks is the flexibility with which features may be defined, particularly not having to worry about their independence. However, there *is* a benefit to working within the stricter, history-based modeling framework: it forces researchers to *think* about the features they are adding.

As a final note related to EM, I mentioned at the end of Chapter 6 how I was using EM to maximize the likelihood of the observed data, but still decoding to maximize the joint likelihood of both observed and hidden data. I also noted that it was intractable to marginalize over the hidden data. However, since marginalization over the hidden data is tractable for a *given* tree (using just the inside algorithm), one decent approximation would be to do k -best parsing with the original decoder, and then perform the inside algorithm for each of the k trees, selecting the tree with the highest probability marginalized over the hidden data. Using the notation of Chapter 6, we want to compute $\arg \max_{T \in \text{TOP}_k} \sum_{T^+} P(T, T^+, S)$, where TOP_k is the set of the k top-scoring augmented trees that have been mapped to their unaugmented versions.

8.3.2 Parameter selection

In Chapter 7, I mentioned how the parameter pruning strategy was somewhat conservative. More aggressive strategies should be explored, to pare down a model to its absolute minimum using Occam’s razor, as it were. In fact, it should be possible to *combine* the EM method of Chapter 6 with the parameter selection method of Chapter 7. If one performs EM on a pruned and therefore more “efficient” model, it is quite possible that there will be less of a chance of EM getting stuck in a lower local maximum, because the hyper-surface being explored would be less “bumpy”. In Chapter 6, I mentioned that an EM-reestimated model was a natural candidate for parse tree re-ranking experiments, such as have been explored by Collins (2000) and Johnson. Along those lines, a pruned model is also a natural candidate for such experiments. Since the pruned model exhibits identical accuracy to the full model even when searching it using a narrower beam, the

suspicion is that for the same k , more “good” parse derivations would exist when k -best decoding using the pruned model as compared to the full model.

8.3.3 Model analysis

More broadly, as ever-more flexible modeling techniques make their way into parsing and other NLP research areas—techniques such as Support Vector Machines that purportedly deal with vast numbers of features in a robust way—the need for model analysis becomes all the more acute. The parameter space is quite different, and the analysis techniques are different than those presented here, but there is a similarity in abstract methodology (*cf.* the feature-selection work of Zhang (2003)). Treating the model as data, performing EDA and determining parameter/feature properties and relationships will provide some of the necessary understanding to propel future research.

Appendix A

Chinese head-finding heuristics

Our default Chinese head-finding heuristics (“head rules”) are shown in Figure A.1.¹ These heuristics are slightly modified from those used by Xia (1999). The BNF description of these rules is as follows:

`<headrule>` ::= (`<parent>` `<instructions>`*)

`<parent>` ::= the parent whose head child is to be found,
or the symbol *, which indicates a default rule

`<instruction>` ::= (`<direction>` `<scanelement>`*)

`<direction>` ::= `l` | `r`

where `l` indicates left-to-right, `r` right-to-left

`<scanelement>` ::= a child nonterminal label to scan for in the specified direction

For each parent, the sequence of instructions is tried in the order specified until a head is found. Instructions of the form (`l`) or (`r`) are default instructions; they pick the leftmost or rightmost child as the head, respectively.

¹A file containing these rules is supplied with the parsing engine, which is available at <http://www.cis.upenn.edu/~dbikel/software.html#stat-parser>; see `src/danbikel/parser/chinese/data/head-rules.lisp`.

(ADJP (r ADJP JJ) (r AD NN CS) (r))
 (ADVP (r ADVP AD) (r))
 (CLP (r CLP M) (r))
 (CP (r DEC SP) (l ADVP CS) (r CP IP) (r))
 (DNP (r DNP DEG) (r DEC) (r))
 (DVP (r DVP DEV) (r))
 (DP (l DP DT) (l))
 (FRAG (r VV NR NN) (r))
 (INTJ (r INTJ IJ) (r))
 (LST (l LST CD OD) (l))
 (IP (r IP VP) (r VV) (r))
 (LCP (r LCP LC) (r))
 (NP (r NP NN NT NR QP) (r))
 (PP (l PP P) (l))
 (PRN (r NP IP VP NT NR NN) (r))
 (QP (r QP CLP CD OD) (r))
 (VP (l VP VA VC VE VV BA LB VCD VSB VRD VNV VCP) (l))
 (VCD (r VCD VV VA VC VE) (r))
 (VRD (r VRD VV VA VC VE) (r))
 (VSB (r VSB VV VA VC VE) (r))
 (VCP (r VCP VV VA VC VE) (r))
 (VNV (r VNV VV VA VC VE) (r))
 (VPT (r VNV VV VA VC VE) (r))
 (UCP (r))
 (WHNP (r WHNP NP NN NT NR QP) (r))
 (WHPP (l WHPP PP P) (l))

Figure A.1: Chinese head-finding rules.

Appendix B

Complete list of parameter classes in Collins' Model 3

This appendix contains tables for all parameter classes in Collins' Model 3, with appropriate modifications and additions from the tables presented in Collins' thesis. The notation is that used throughout Chapter 3. In particular, for notational brevity we use $M(w, t)_i$ to refer to the three items M_i , t_{M_i} and w_{M_i} that constitute some fully-lexicalized modifying nonterminal, and similarly $M(t)_i$ to refer to the two items M_i and t_{M_i} that constitute some partially-lexicalized modifying nonterminal. The (unlexicalized) nonterminal-mapping functions α and γ are defined at the beginning of §3.5. As a shorthand, $\gamma(M(t)_i) = \gamma(M_i), t_{M_i}$.

The head-generation parameter class, P_H , gap-generation parameter class, P_G and subcat-generation parameter classes, P_{subcat_L} and P_{subcat_R} , have back-off structures as follows.

Back-off level	$P_H(H \dots)$	$P_G(G \dots)$ $P_{subcat_L}(subcat_L \dots)$ $P_{subcat_R}(subcat_R \dots)$
0	$\gamma(P), w_h, t_h$	$\alpha(\gamma(P)), \alpha(\gamma(H)), w_h, t_h$
1	$\gamma(P), t_h$	$\alpha(\gamma(P)), \alpha(\gamma(H)), t_h$
2	$\gamma(P)$	$\alpha(\gamma(P)), \alpha(\gamma(H))$

The two parameter classes for generating modifying nonterminals that are not dominated by a base NP, P_M and P_{M_w} , have the following back-off structures. Recall that back-off level 2 of the P_{M_w} parameters includes words that are the heads of the observed roots of sentences (that is, the head word of the entire sentence).

Back-off level	$P_M(M(t)_i, \text{coord}, \text{punc} \dots)$
0	$\alpha(P), \gamma(H), w_h, t_h, \Delta_{side}, \text{subcat}_{side}, \text{side}$
1	$\alpha(P), \gamma(H), t_h, \Delta_{side}, \text{subcat}_{side}, \text{side}$
2	$\alpha(P), \gamma(H), \Delta_{side}, \text{subcat}_{side}, \text{side}$

Back-off level	$P_{M_w}(w_{M_i} \dots)$
0	$\gamma(M(t)_i), \text{coord}, \text{punc}, \alpha(P), \gamma(H), w_h, t_h, \Delta_{side}, \text{subcat}_{side}, \text{side}$
1	$\gamma(M(t)_i), \text{coord}, \text{punc}, \alpha(P), \gamma(H), t_h, \Delta_{side}, \text{subcat}_{side}, \text{side}$
2	t_{M_i}

The two parameter classes for generating modifying nonterminals that are children of base NPs (NPB nodes), $P_{M,\text{NPB}}$ and $P_{M_w,\text{NPB}}$, have the following back-off structures. Back-off level 2 of the $P_{M_w,\text{NPB}}$ parameters includes words that are the heads of the observed roots of sentences (that is, the head word of the entire sentence). Also, note that there is no `coord` flag, as coordinating conjunctions are generated like regular modifying nonterminals when they are dominated by NPB. Finally, we define $M_0 = H$, that is, the head nonterminal label of the base NP that was generated using a P_H parameter.

Back-off level	$P_{M,\text{NPB}}(M(t)_i, \text{punc} \dots)$	$P_{M_w,\text{NPB}}(w_{M_i} \dots)$
0	$P, M(w, t)_{i-1}, \text{side}$	$M_i, t_{M_i}, \text{punc}, P, M(w, t)_{i-1}, \text{side}$
1	$P, M(t)_{i-1}, \text{side}$	$M_i, t_{M_i}, \text{punc}, P, M(t)_{i-1}, \text{side}$
2	P, M_{i-1}, side	t_{M_i}

The two parameter classes for generating punctuation and coordinating conjunctions, P_{punc} and P_{coord} , have the following back-off structures (Collins, p.c.), where

- *type* is a flag that obtains the value `p` in the history contexts of P_{punc} parameters and `c` in the

history contexts of P_{coord} parameters,

- $M(w, t)_i$ is the modifying preterminal that is being conjoined to the head child,
- t_p/t_c is the particular preterminal (part of speech tag) that is conjoining the modifier to the head child (such as “CC” or “:”) and
- w_p/w_c is the particular word that is conjoining the modifier to the head child (such as “and” or “:”).

Back-off level	$P_{coord}(t_c \dots)$ $P_{punc}(t_p \dots)$	$P_{coord_w}(w_c \dots)$ $P_{punc_w}(w_p \dots)$
0	$w_h, t_h, P, H, M(w, t)_i, type$	$t_{type}, w_h, t_h, P, H, M(w, t)_i, type$
1	$t_h, P, H, M(t)_i, type$	$t_{type}, t_h, P, H, M(t)_i, type$
2	$type$	t_{type}

The parameter classes for generating fully-lexicalized root nonterminals given the hidden root +TOP+, P_{TOP} and P_{TOP_w} , have the following back-off structures (identical to the table in Table 3.3).

Back-off level	$P_{TOP_{NT}}(H(t) \dots)$	$P_{TOP_w}(w \dots)$
0	+TOP+	$t, H, +TOP+$
1	n/a	t

The parameter classes for generating marginal probabilities on lexicalized nonterminals $M(w, t)$, P_{prior_w} and $P_{prior_{NT}}$, have the following back-off structures, where *prior* is a dummy variable to indicate that P_{prior_w} is not smoothed (although the P_{prior_w} parameters still have an associated smoothing weight; see §3.5.7.2, footnote).

Back-off level	$P_{prior_w}(w, t \dots)$	$P_{prior_{NT}}(M \dots)$
0	<i>prior</i>	w, t
1	<i>prior</i>	t

References

- Alshawi, Hiyan and David Carter. 1994. Training and scaling preference functions for disambiguation. *Computational Linguistics*, 23(4):635–648.
- Bak, Per. 1996. *How Nature Works*. Copernicus.
- Baker, J. K. 1979. Trainable grammars for speech recognition. In *Spring Conference of the Acoustical Society of America*, pages 547–550, Boston, MA.
- Bies, A. 1995. Bracketing guidelines for Treebank II style Penn Treebank Project. <ftp://ftp.cis.upenn.edu/pub/treebank/doc/manual/root.ps.gz>.
- Bikel, Daniel M. 2000. A statistical model for parsing and word-sense disambiguation. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, Hong Kong, October.
- Bikel, Daniel M. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of HLT2002*, San Diego, CA.
- Bikel, Daniel M. and David Chiang. 2000. Two statistical parsing models applied to the Chinese Treebank. In Martha Palmer, Mitch Marcus, Aravind Joshi, and Fei Xia, editors, *Proceedings of the Second Chinese Language Processing Workshop*, pages 1–6, Hong Kong.
- Bikel, Daniel M., Richard Schwartz, Ralph Weischedel, and Scott Miller. 1997. Nymble: A high-performance learning name-finder. In *Fifth Conference on Applied Natural Language Processing*, pages 194–201, Washington, D.C.

- Black, Ezra, Frederick Jelinek, John Lafferty, David Magerman, Robert Mercer, and Salim Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*, Harriman, New York.
- Black, Ezra, John Lafferty, and Salim Roukos. 1992. Development and evaluation of a broad-coverage probabilistic grammar of english-language computer manuals. In *Proceedings of the 30th ACL*, pages 185–192.
- Bod, Rens. 1992. A computational model of language performance: Data oriented parsing. In *Proceedings COLING-92*, Nantes, France.
- Bod, Rens. 1993. Using an annotated language corpus as a virtual stochastic grammar. In *Proceedings AAAI'93*, Washington, D.C.
- Booth, T. L. and R. A. Thompson. 1973. Applying probability measures to abstract languages. In *IEEE Transactions on Computers*, volume C-22, pages 442–450.
- Charniak, Eugene. 1996. Tree-bank grammars. Technical report, Department of Computer Science, Brown University.
- Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Menlo Park. AAAI Press/MIT Press.
- Charniak, Eugene. 2000. A maximum entropy-inspired parser. In *Proceedings of the 1st NAACL*, pages 132–139, Seattle, Washington, April 29 to May 4.
- Chen, John and K. Vijay-Shanker. 2000. Automated extraction of TAGs from the Penn Treebank. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 65–76, Trento.
- Chiang, David. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

- Chiang, David and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of COLING'02*, Taipei, Taiwan.
- Cohen, Paul R. 1995. *Empirical Methods for Artificial Intelligence*. MIT Press.
- Collins, Michael. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191.
- Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL-EACL '97*, pages 16–23.
- Collins, Michael. 2000. Discriminative reranking for natural language parsing. In *International Conference on Machine Learning*.
- Collins, Michael and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL-02*, pages 263–270, Philadelphia, Pennsylvania.
- Collins, Michael John. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.
- Cover, Thomas and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons, Inc., New York.
- Daelemans, Walter, Jakub Zabrel, and Ko van der Sloot. 2002. TiMBL: Tilburg memory-based learner. Technical report, Induction of Linguistic Knowledge, Tilburg University, November.
- Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc. B*, 39:1–38.
- Eisner, Jason. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, August.

- Fellbaum, Christiane, Joachim Grabowski, and Shari Landes. 1998. Performance and confidence in a semantic annotation task. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Massachusetts, chapter 9.
- Francis, W. N. and H. Kučera. 1979. *Manual of Information to accompany A Standard Corpus of Present-Day Edited American English, for use with Digital Computers*. Department of Linguistics, Brown University, Providence, Rhode Island.
- Gildea, Daniel. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, Pittsburgh, Pennsylvania.
- Gildea, Daniel and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *Proceedings of ACL 2000*, Hong Kong.
- Gildea, Daniel and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of ACL 2002*, Philadelphia, Pennsylvania.
- Goodman, Joshua. 1997a. Global thresholding and multiple-pass parsing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*.
- Goodman, Joshua. 1997b. Probabilistic feature grammars. In *Proceedings of the International Workshop on Parsing Technologies*, September.
- Goodman, Joshua T. 2001. A bit of progress in language modeling. *Computer Speech & Language*, pages 403–434.
- Hardy, G. H., J. E. Littlewood, and G. Pólya. 1988. *Inequalities*. Cambridge University Press, 2nd edition.
- Henderson, John C. and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of the Fourth Conference on Empirical Methods in Natural Language Processing*, College Park, Maryland.
- Hwa, Rebecca. 1998. An empirical evaluation of probabilistic lexicalized tree insertion grammars. In *Proceedings of ACL-COLING 1998*, pages 557–563.

- Hwa, Rebecca. 2001. On minimizing training corpus for parser acquisition. In *Proceedings of the Fifth Computational Natural Language Learning Workshop*, July.
- Hwa, Rebecca, Philip Resnik, and Amy Weinberg. 2002. Breaking the resource bottleneck for multilingual parsing. In *Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data, Third International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Canary Islands, Spain, June.
- Jelinek, Frederick. 1997. *Statistical Methods for Speech Recognition*. The MIT Press.
- Johnson, Mark. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:613–632.
- Joshi, Aravind K. and Yves Schabes. 1997. Tree-adjoining grammars. In A. Salomaa and G. Rosenberg, editors, *Handbook of Formal Languages and Automata*, volume 3. Springer-Verlag, Heidelberg, pages 69–124.
- Kingsbury, Paul and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Spain.
- Klein, Dan and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics*.
- Klein, Dan and Christopher D. Manning. 2002a. Conditional structure versus conditional estimation in NLP models. In *Proceedings of the 2002 Conference on Empirical Methods for Natural Language Processing*.
- Klein, Dan and Christopher D. Manning. 2002b. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*.
- Kroch, Anthony. 2000. Verb-object order in Early Middle English. In *Proceedings of the Fifth Diachronic Generative Syntax Conference (DIGS 5)*, University of York.

- Lafferty, John, Fernando Pereira, and Andrew McCallum. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.
- Lafferty, John, Daniel Sleator, and Davy Temperley. 1992. Grammatical trigrams: A probabilistic model of LINK grammar. In *Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*. Cambridge, MA, 1992, pages 89–97, Menlo Park, CA. AAAI Press.
- 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.
- Lee, Lillian. 1999. Measures of distributional similarity. In *Proceedings of the 37th ACL*, pages 25–32.
- Levy, Roger and Christopher D. Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank? In *Proceedings of the 23rd Annual Meeting of the Association of Computational Linguistics*, pages 439–446.
- Lin, Jianhua. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.
- Magerman, D. 1995. Statistical decision tree models for parsing. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, Massachusetts. Morgan Kaufmann Publishers.
- Magerman, D. and M. Marcus. 1991. Parsing the voyager domain using pearl. In *Speech and Natural Language Workshop*, pages 231–236, Pacific Grove, California. Morgan Kaufmann Publishers.
- Magerman, David. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, University of Pennsylvania, Philadelphia, Pennsylvania.
- Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

- Miller, Scott, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 1998. SIFT – Statistically-derived Information From Text. In *Seventh Message Understanding Conference (MUC-7)*, Washington, D.C.
- Palmer, Martha, Hoa Trang Dang, and Christiane Fellbaum. 2004. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*.
- Pereira, F. and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135.
- Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, California.
- Rambow, Owen, K. Vijay-Shanker, and David Weir. 1995. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 151–158.
- Ratnaparkhi, Adwait. 1996. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing*, May.
- Ratnaparkhi, Adwait. 1997. A simple introduction to maximum entropy models for natural language processing. Technical Report IRCS Report 97–08, Institute for Research in Cognitive Science, May.
- Resnik, Philip. 1992. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of COLING-92*, pages 418–424.
- Sarkar, Anoop. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the Second Conference of the North American Association of Computational Linguistics (NAACL-2001)*, pages 175–182, Pittsburgh, PA.
- Sarkar, Anoop. 2002. *Combining Labeled and Unlabeled Data in Statistical Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

- Schabes, Yves. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, University of Pennsylvania.
- Schabes, Yves. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proceedings of COLING-92*, pages 426–432.
- Schabes, Yves and Richard C. Waters. 1995. Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21:479–513.
- Sekine, S., J. Carroll, S. Ananiadou, and J. Tsujii. 1992. Automatic learning for semantic collocation. In *Proceedings of the Third Conference on Applied Natural Language Processing*.
- Slonim, Noam, Nir Friedman, and Naftali Tishby. 2002. Unsupervised document classification using sequential information maximization. Technical Report 2002–19, Leibniz Center, The School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel.
- Stolcke, Andreas. 1998. Entropy-based pruning of backoff language models. In *Proceedings DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274, Lansdowne, VA.
- Sun, Honglin and Daniel Jurafsky. 2004. Shallow semantic parsing of Chinese. In *Proceedings of HLT-NAACL 2004*, pages 249–256, Boston, Massachusetts, U.S.A., May 2–7.
- Thomason, Michael G., 1986. *Handbook of Pattern Recognition and Image Processing*, chapter 5. Harcourt Brace Jovanovich.
- Vijay-Shanker, K. and David Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.
- Weischedel, R., M. Meteer, R. Schwartz, L. Ramshaw, and J. Palmucci. 1993. Coping with ambiguity and unknown words through probabilistic methods. *Computational Linguistics*, 19(2):359–382.

- Witten, I. T. and T. C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. In *IEEE Transactions on Information Theory*, volume 37, pages 1085–1094, July.
- Xia, Fei. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*.
- Xia, Fei, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Shizhe Huang, Tony Kroch, and Mitch Marcus. 2000. Developing Guidelines and Ensuring Consistency for Chinese Text Annotation. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.
- Zhang, H. H. 2003. Variable selection for SVM via basis pursuit in smoothing spline ANOVA models. Technical Report 2257, Institute of Statistics Mimeo Series, North Carolina State University.