Intricacies of Collins' Parsing Model

Daniel M. Bikel*
University of Pennsylvania

This article documents a large set of heretofore unpublished details Collins used in his parser, such that, along with Collins' (1999) thesis, this article contains all information necessary to duplicate Collins' benchmark results. Indeed, these as-yet-unpublished details account for an 11% relative increase in error from an implementation including all details to a clean-room implementation of Collins' model. We also show a cleaner and equally well-performing method for the handling of punctuation and conjunction and reveal certain other probabilistic oddities about Collins' parser. We not only analyze the effect of the unpublished details, but also reanalyze the effect of certain well-known details, revealing that bilexical dependencies are barely used by the model and that head choice is not nearly as important to overall parsing performance as once thought. Finally, we perform experiments that show that the true discriminative power of lexicalization appears to lie in the fact that unlexicalized syntactic structures are generated conditioning on the headword and its part of speech.

1. Introduction

Michael Collins' (1996, 1997, 1999) parsing models 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 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 **multilingual 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' (1999) thesis, we reproduced all the parameters described but did not achieve nearly the same high performance on the well-established development test set of Section 00 of

* Department of Computer and Information Science, 3330 Walnut Street, Philadelphia, PA 19104. E-mail: dbikel@linc.cis.upenn.edu

1 This engine is publicly available at `http://www.cis.upenn.edu/~dbikel/software.html`

Submission received: 18 January 2003; Revised submission received: 20 March 2004; Accepted for publication: 10 June 2004

the Penn Treebank. Together with Collins' thesis, this article contains all the information necessary to replicate Collins' parsing results.[2] Specifically, this article 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 article also discusses the rationale for various parameter choices. In general, we will limit our discussion to Collins' Model 2, but we make occasional reference to Model 3, as well.

2. 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 motivation is related to the first: science dictates that experiments be replicable, for this 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 "plateau" in performance. Rather than simply throwing features into a new model and measuring their effect in a crude way using standard evaluation metrics, this work aims to provide a more thorough understanding of the *nature* of a model's features. This understanding not only is 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.

2 In 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) reimplemented Collins' Model 1 but obtained results with roughly 16.7% more relative error than Collins' reported results using that model.

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

3. 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 headword (and that headword's part of speech) that the node dominates. The lexicalized PCFG that sits behind Model 2 has rules of the form

$$P \rightarrow L_n L_{n-1} \cdots L_1 H R_1 \cdots R_{n-1} R_n \tag{1}$$

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, because it includes the distance and subcategorization frame features, the model captures a crucial bit of linguistic reality, namely, 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 7).

4. 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 11 preprocessing steps necessary to prepare training trees when using Collins' parsing model:

1. pruning of unnecessary nodes
2. adding base NP nodes (`NPB`s)
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 of presentation in the foregoing list 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 required 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.

4.1 Coordinated Phrases

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

- it has a nonhead child that is a coordinating conjunction and
- that conjunction is either
  - posthead but nonfinal, or
  - immediately prehead but noninitial (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 in which the head-child is truly conjoined to some other phrase, as opposed to a phrase in which, say, there is an initial `CC`, such as an `S` that begins with the conjunction *but*.

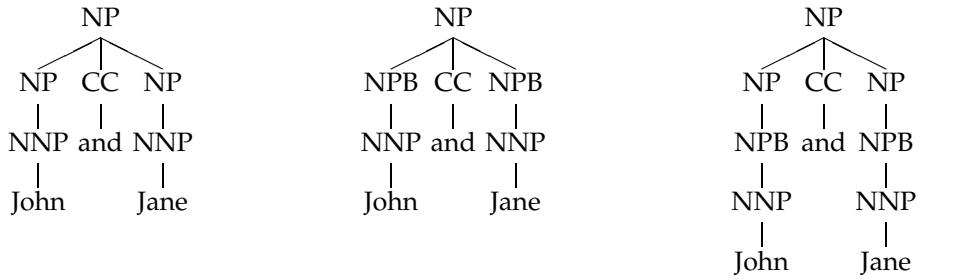
4.2 Pruning of Unnecessary Nodes

As a preprocessing step, pruning of unnecessary nodes 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: First, 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 second, neither of these types of preterminals—nor *any* punctuation marks, for that matter—counts towards the parsing score.

4.3 Adding Base NP Nodes

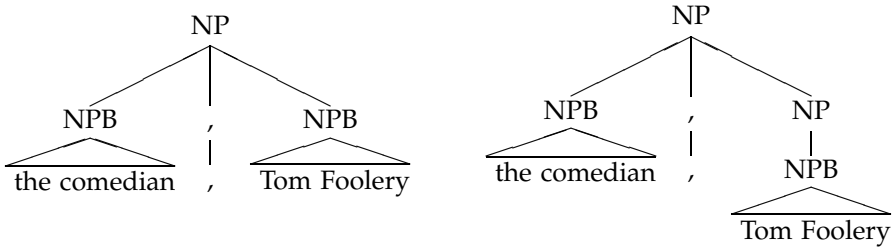
An NP is **basal** when it does not itself dominate an NP; such `NP` nodes are relabeled `NPB`. More accurately, an NP is basal when it dominates no other NPs except possessive NPs, where a possessive NP is an NP that dominates `POS`, the preterminal possessive

4 Our positional descriptions here, such as "posthead but nonfinal," refer to positions within the list of immediately dominated children of the coordinated phrase node, as opposed to positions within the entire sentence.



(a) Coordinated phrase (b) Base NPs relabeled (c) Extra NP nodes inserted

Figure 1
An NP that constitutes a coordinated phrase.



(a) Before extra NP addition (the NPB *the comedian* is the head child).

(b) After extra NP insertion.

Figure 2
A nonhead NPB child of NP requires insertion of extra NP.

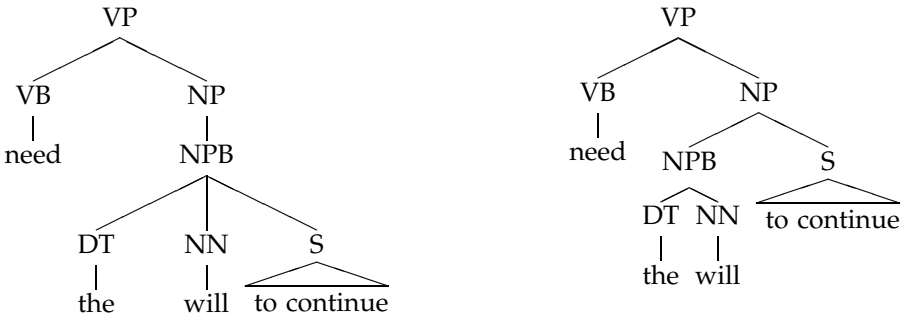
marker for the Penn Treebank. These possessive NPs are almost always themselves base NPs and are therefore (almost always) relabeled NPB.

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 is described in Collins' thesis, however. The extra NP level is added if one of the following conditions holds:

- The parent of the NPB is not an NP.
- The parent of the NPB *is* an NP but constitutes a coordinated phrase (see Figure 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 2).[5]

In postprocessing, 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.

5 Only applicable if relabeling of NPs is performed using a preorder tree traversal.



(a) Before repair. (b) After repair.

Figure 3
An NPB is "repaired."

4.4 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).[6] While such a transformation is reasonable, it is interesting to note that Collins' parser performs no equivalent detransformation when parsing is complete, meaning that when the parser produces the "repaired" structure during testing, there is a spurious NP bracket.[7]

4.5 Adding Gap Information

The gap feature is discussed extensively in chapter 7 of Collins' thesis and is applicable only to his Model 3. The preprocessing step in which gap information is added 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 threaded-gap feature is represented by appending -g to every node label in the chain. The only detail we would like to highlight here is that an implementation of this preprocessing step should check for cases in which threading is impossible, such as when two filler-gap dependencies cross. An implementation should be able to handle nested filler-gap dependencies, however.

4.6 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 phrases. Collins' example of

$$[_{S}\,[_{S}\,\texttt{Flying planes}]\,\texttt{is dangerous}\,]$$

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

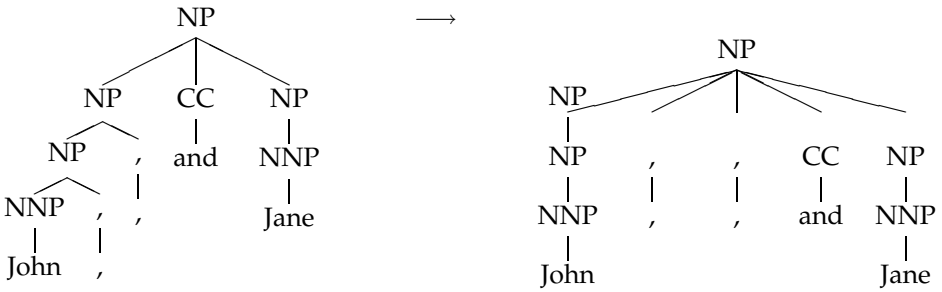


Figure 4
Raising punctuation: Perverse case in which multiple punctuation elements appear along a frontier of a subtree.

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 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 Sections 4.9 and 4.11)

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]

4.7 Removing Null Elements
Removing null elements simply involves pruning the tree to eliminate any subtree that dominates only null elements. The special trace tag that is inserted in the step that adds gap information (Section 4.5) is excluded, as it is specifically chosen to be something other than the null-element preterminal marker (which is -NONE- in the Penn Treebank).

4.8 Raising Punctuation
The step in which punctuation is raised is discussed in detail in chapter 7 of Collins' thesis. The main idea is to raise punctuation—which is any preterminal subtree in which 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," that is, pruned. In addition, any implementation of this step should handle the case in which multiple punctuation elements appear as the initial or final children of some node, as well as the more pathological case in which multiple punctuation elements appear along the left or right frontier of a subtree (see Figure 4). Finally, it is not clear what to do with nodes that dominate *only* punctuation preterminals. Our implementation simply issues a warning in such cases and leaves the punctuation symbols untouched.

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

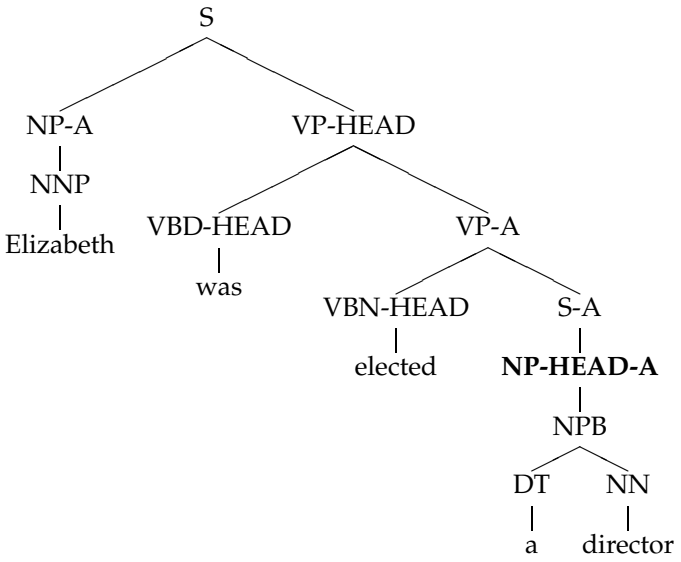


Figure 5
Head-children are not exempt from being relabeled as arguments.

4.9 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 Section 4.1) are exempt from being relabeled as argument nonterminals.
- 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 5).[9]

4.10 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.

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

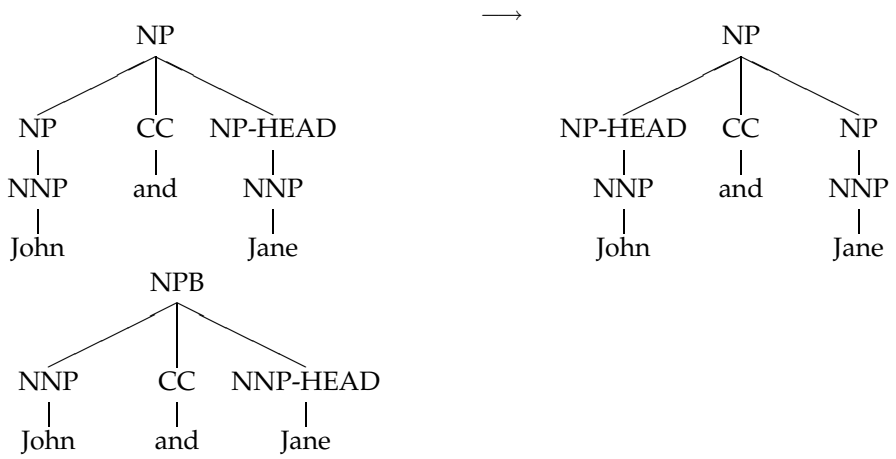


Figure 6
Head moves from right to left conjunct in a coordinated phrase, *except* when the parent nonterminal is NPB.

4.11 Repairing Subjectless Sentences

With arguments identified as described in Section 4.9, if a subjectless sentence is found to have an argument prior to its head, this step detransforms the SG so that it reverts to being an S.

4.12 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.

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 in which 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 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* as though it constitutes a coordinated phrase—it has a CC that is noninitial but to the left of the currently chosen head—but the currently chosen head should remain chosen.[11] As we shall see, there is exceptional behavior for base NPs in almost every part of the Collins parser.

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

11 In Section 4.1, 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 noninitial 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.

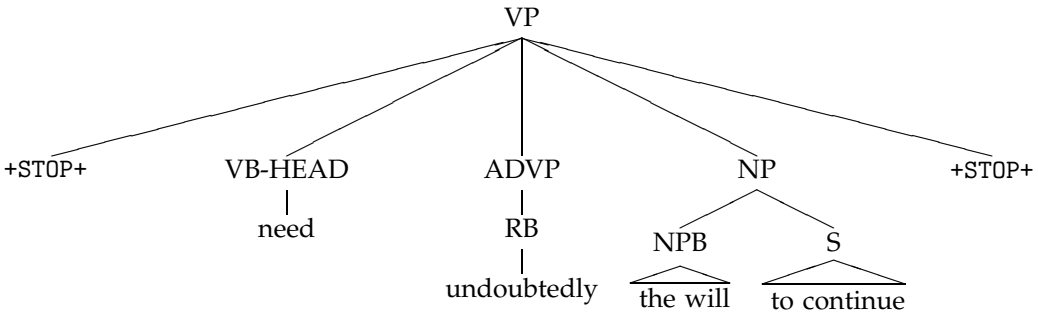


Figure 7
`vi` feature is `true` when generating right-hand `+STOP+` nonterminal, because the NP *the will to continue* contains a verb.

5. Training

The trainer's job is to decompose annotated training trees into a series of head- and modifier-generation steps, recording the counts of each of these steps. Referring to (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 is counted. Even with all this decomposition, sparse data are 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.

5.1 Verb Intervening

As mentioned in Section 3, 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 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] More formally, 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:

$$\texttt{cv}(P) = \begin{cases} \bigvee_{M \text{ child of } P} \texttt{cv}(M) & \text{if } M \text{ is not a preterminal} \\ \texttt{true} & \text{if } P \text{ is a verb preterminal} \\ \texttt{false} & \text{otherwise} \end{cases} \quad (2)$$

12 Note that *any* word 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—that is, anything in the *history*—can appear in the conditioning context.

Referring to (2), we define the verb-intervening predicate recursively on the first-order Markov process generating modifying nonterminals:

$$\texttt{vi}(L_i) = \begin{cases} \texttt{false} & \text{if } i \leq 1 \\ \texttt{cv}(L_{i-1}) \vee \texttt{vi}(L_{i-2}) & \text{if } i > 1 \end{cases} \tag{3}$$

and similarly for right modifiers.

What is considered to be a verb? While this is not spelled out, as it happens, a verb 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 $\texttt{cv}(\texttt{NPB}) = \texttt{false}$.

5.2 Skip Certain Trees

One oddity of Collins' trainer that we mention here for the sake of completeness is that it skips certain training trees. For "odd historical reasons,"[13] the trainer skips all trees with more than 500 tokens, where a token is considered in this context 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 Treebank, 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 Table 5) when these trees are included.

5.3 Unknown Words

5.3.1 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 (page 186)." 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 six, not five.[14] To be absolutely unambiguous, words that occur *fewer than six times*, which is to say, words that occur *five times or fewer,* in the data are considered "unknown."

5.3.2 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

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

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

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 bilexical 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 Section 6.9.2.

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

6. 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 explore smoothing in detail in Section 6.8).

6.1 Mapped Versions of the Set of Nonterminals

In Sections 4.5 and 4.9, 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 denote the "argument removal" mapping function as alpha and the "gap removal" mapping function as gamma. For example:

- $\alpha(\texttt{NP-A-g}) = \texttt{NP-g}$
- $\gamma(\texttt{NP-A-g}) = \texttt{NP-A}$
- $\alpha(\gamma(\texttt{NP-A-g})) = \texttt{NP}$

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

6.2 The Head Parameter Class

The head nonterminal is generated conditioning on its parent nonterminal label, as well as the headword 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 denote the parameter class as follows:

$$P_H(H \mid \gamma(P), w_h, t_h) \tag{4}$$

6.3 The Subcategorization 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:

$$P_{subcat_L}(subcat_L \mid \alpha(\gamma(H)), \alpha(\gamma(P)), w_h, t_h) \tag{5}$$

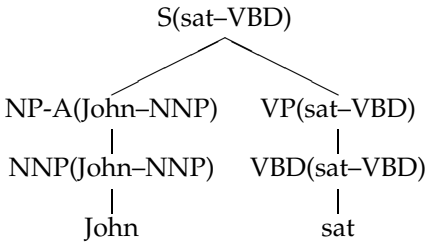


Figure 8
A fully lexicalized tree. The VP node is the head-child of S.

$$P_{subcat_R}(subcat_R | \alpha(\gamma(H)), \alpha(\gamma(P)), w_h, t_h) \quad (6)$$

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 8, after the root node of the tree has been generated (see Section 6.10), the head child VP is generated, conditioning on both the parent label S and the headword 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 items 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.

6.4 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 1. A fully lexicalized nonterminal has three components: the nonterminal label, the headword, and the headword's part of speech. Fully lexicalized modifying nonterminals are generated 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. In the

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

16 Our 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 runtime.

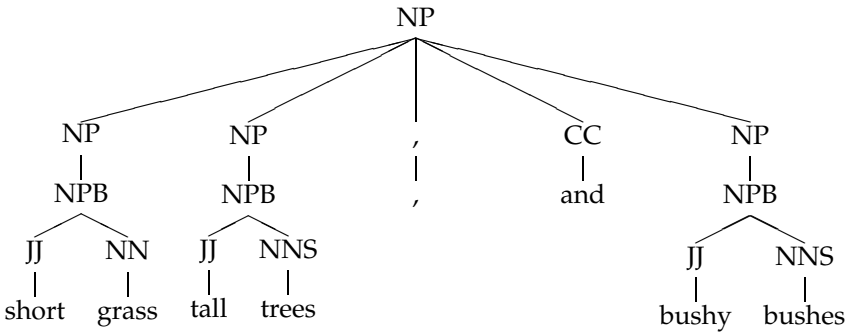


Figure 9
A tree containing both punctuation and conjunction.

first step, a **partially lexicalized** version of the nonterminal is generated, consisting of the unlexicalized label plus the part of speech of its headword. These partially lexicalized modifying nonterminals are generated conditioning on the parent label, the head label, the headword, the head tag, the current state of the dynamic subcat, and a distance metric. Symbolically, the parameter classes are

$$P_L(L(t)_i \mid \alpha(P), \gamma(H), w_h, t_h, subcat_L, \Delta_L) \tag{7}$$

$$P_R(R(t)_i \mid \alpha(P), \gamma(H), w_h, t_h, subcat_R, \Delta_R) \tag{8}$$

where Δ denotes the distance metric.[17] As discussed above, one of the two components of this distance metric is the `vi` 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 headword itself, where, because of the chain rule, the conditioning context consists of everything in the histories of expressions (7) and (8) plus the partially lexicalized modifier. As there are some interesting idiosyncrasies with these headword-generation parameters, we describe them in more detail in Section 6.9.

6.5 The Punctuation and Coordinating Conjunction Parameter Classes

6.5.1 Inconsistent Model. As discussed in Section 4.8, 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 "conjoins" 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 coordinated phrase in Section 4.1, conjunction via a `CC` node or a punctuation node always occurs posthead (i.e., as a right-sibling of the head). Put another way, if a conjunction or punctuation mark occurs prehead, it is

17 Throughout this article we use the notation $L(w,t)_i$ to refer to the three items that constitute a fully lexicalized left-modifying nonterminal, which are the unlexicalized label L_i, its headword 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.

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

not generated via this mechanism.[19] Furthermore, even 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 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 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 headword 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 headwords of the two conjoined phrases, and the intervening `CC` is similarly generated. A simplified version of the probability of generating all these children is summarized as follows:

$$\begin{aligned}
&\hat{p}_H(\texttt{NP} \mid \texttt{NP,grass,NN}) \cdot \\
&\hat{p}_R(\texttt{NP(trees,NNS),punc=0,coord=0} \mid \texttt{NP,NP,grass,NN}) \cdot \\
&\hat{p}_R(\texttt{NP(bushes,NNS),punc=1,coord=1} \mid \texttt{NP,NP,grass,NN}) \cdot \\
&\hat{p}_{punc}(\texttt{,(,)} \mid \texttt{NP,NP,NP,bushes,NNS,grass,NN}) \cdot \\
&\hat{p}_{CC}(\texttt{CC(and)} \mid \texttt{NP,NP,NP,bushes,NNS,grass,NN})
\end{aligned} \tag{9}$$

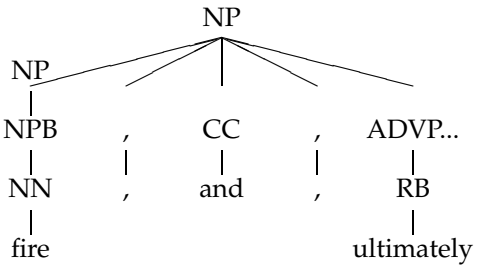
The idea is that using the chain rule, the generation of two conjuncts and that which conjoins them is estimated as one large joint event.[20]

This scheme of using flags to trigger the P_{punc} and P_{CC} parameters is problematic, at least from a theoretical standpoint, as it causes the model to be inconsistent. Figure 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, nonzero probability, the sum $\sum_T P(T)$, where T is a possible tree generated by the model, diverges, meaning the model is inconsistent (Booth and Thompson 1973). Another consequence of not generating posthead conjunctions and punctuation as first-class words is that they

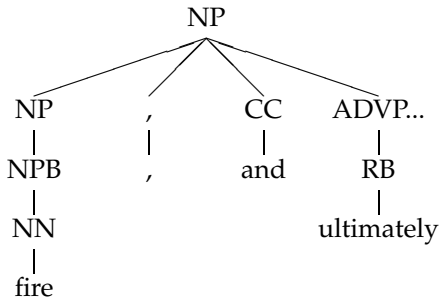
19 In fact, if punctuation occurs before the head, it is not generated at all—a deficiency in the parsing model that appears to be a holdover from the deficient punctuation handling in the model of Collins (1997).

20 In (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.

(a)



(b)



(c)

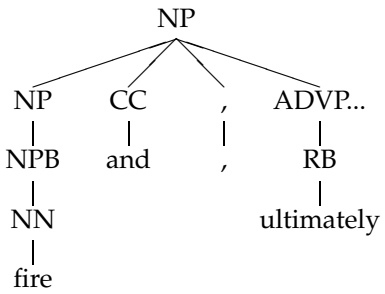


Figure 10
The Collins model assigns equal probability to these three trees.

do not count when calculating the head-adjacency component of 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.

6.5.2 History Mechanism. In our emulation of Collins' model, we use the *history*, rather than the dedicated parameter classes P_{CC} and P_{punc}, to estimate the joint event of generating a conjunction (or punctuation mark) and its two conjuncts. The first big change that results is that we treat punctuation preterminals and `CC`s as first-class objects, meaning that they are generated in the same way as 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

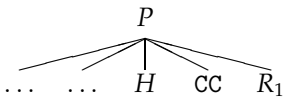
mapping function:

$$\delta(M_i) = \begin{cases} \texttt{+START+} & \text{if } i = 0 \\ \texttt{CC} & \text{if } M_i = \texttt{CC} \\ \texttt{+PUNC+} & \text{if } M_i = \texttt{,} \text{ or } M_i = \texttt{:} \\ \texttt{+OTHER+} & \text{otherwise} \end{cases} \tag{10}$$

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

$$P_M\left(M(t)_i \mid \alpha(P), \gamma(H), w_h, t_h, subcat_{side}, \texttt{vi}(M_i), \delta(M_{i-1}), side\right) \tag{11}$$

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 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 delta function) *and* achieved an estimate of the joint event of a conjunction and its conjuncts, albeit with different dependencies, that is, 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 this:

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

whereas under the new parameterization, it looks like this:

$$\hat{p}_H(H \mid P) \cdot \hat{p}_R(\texttt{CC} \mid P, H, \texttt{+START+}) \cdot \hat{p}_R(R_1 \mid P, H, \texttt{CC})$$

Either way, the probability of the joint conditional event $\{H, \texttt{CC}, R_1 \mid P\}$ is being estimated, but with the new method, there is no need to add two new specialized parameter classes, and the new method does not introduce inconsistency into the model. Using less simplification, the probability of generating the five children of Figure 9 is now

$$\begin{aligned} &\hat{p}_H(\texttt{NP} \mid \texttt{NP,grass,NN}) \cdot \\ &\hat{p}_M(\texttt{NP(trees,NNS)} \mid \texttt{NP,NP,grass,NN,\{\},false,+START+,right}) \cdot \\ &\hat{p}_M(\texttt{,(,,,)} \mid \texttt{NP,NP,grass,NN,\{\},false,+OTHER+,right}) \cdot \\ &\hat{p}_M(\texttt{CC(and,CC)} \mid \texttt{NP,NP,grass,NN,\{\},false,+PUNC+,right}) \cdot \\ &\hat{p}_M(\texttt{NP(bushes,NNS)} \mid \texttt{NP,NP,grass,NN,\{\},false,CC,right}) \end{aligned} \tag{12}$$

21 Originally, we 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 Section 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 article are with the original model.

As shown in Section 8.1, this new parameterization yields virtually identical performance to that of the Collins model.[22]

6.6 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 always generated conditioning not on the head, but on the previously generated modifier. That is, we modify expressions (7) and (8) to be

$$P_{L,\texttt{NPB}}(L(t)_i \mid P, L(w,t)_{i-1}) \tag{13}$$

$$P_{R,\texttt{NPB}}(R(t)_i \mid P, R(w,t)_{i-1}) \tag{14}$$

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 Section 4.12), 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 crucially, the modifier is always conjoined (via the punctuation mark) to the "pseudohead" that is the previously generated modifier. Consequently, when some right modifier R_i is generated, 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., nonpunctuation) preterminal.[24]

Base NPs are also exceptional with respect to determining chart item equality, the comma-pruning rule, and general beam pruning (see Section 7.2 for details).

6.7 Parameter Classes for Priors on Lexicalized Nonterminals

Two parameter classes that make their appearance only in Appendix E of Collins' thesis are those that compute priors on lexicalized nonterminals. These priors are used as a crude proxy for the outside probability of a chart item (see Baker [1979] and Lari and Young [1990] for full descriptions of the Inside–Outside algorithm). Previous work (Goodman 1997) 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 prior on the root (lexicalized) nonterminal label of the derivation forest represented by a particular chart item is used for this purpose in Collins' parser.

22 As described in Bikel (2002), 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 delta and the transition function tau presented in this section are just two examples of this capability.

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

24 Interestingly, unlike in 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.

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

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

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

6.8 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 smoothes the distributions based on full contexts with those from 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 smoothes $P_{H_0}(H \mid P, w_h, t_h)$ with $P_{H_1}(H \mid P, t_h)$ and $P_{H_2}(H \mid P)$. For some conditional probability $p(A \mid B)$, let us call the reduced context at the ith back-off level $\phi_i(B)$, where typically $\phi_0(B) = B$. Each estimate in the back-off chain is computed via maximum-likelihood (ML) estimation, and the overall smoothed estimate with n back-off levels is computed using $n-1$ smoothing weights, denoted $\lambda_0, \ldots, \lambda_{n-2}$. These weights are used in a recursive fashion: The smoothed version $\tilde{e}_i = \tilde{p}_i(A \mid \phi_i(B))$ of an unsmoothed ML estimate $e_i = \hat{p}_i(A \mid \phi_i(B))$ at back-off level i is computed via the formula

$$\tilde{e}_i = \lambda_i e_i + (1 - \lambda_i)\tilde{e}_{i+1}, \quad 0 \leq i < n-1,\ \tilde{e}_{n-1} = e_{n-1} \tag{15}$$

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

$$\tilde{e}_0 = \lambda_0 e_0 + (1 - \lambda_0)\left[\lambda_1 e_1 + (1 - \lambda_1)e_2\right] \tag{16}$$

It is easy to prove by structural induction that if

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

then

$$\sum_A \tilde{p}_0(A \mid \phi_0(B)) = 1 \tag{17}$$

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.

6.8.1 Deficient Model. As previously 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 nonzero) "probability" of 10^{-19}, resulting in smoothed estimates of the form

$$\tilde{e}_0 = \lambda_0 e_0 + (1 - \lambda_0)\left[\lambda_1 e_1 + (1 - \lambda_1)\left[\lambda_2 e_2 + (1 - \lambda_2) \cdot 10^{-19}\right]\right] \tag{18}$$

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 (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).[25]

6.8.2 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.[26] The multiplicative constant five is used to give less weight to the back-off levels with more context and 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

$$\lambda_i = \begin{cases} \frac{c_i}{c_i + f_t + f_f u_i} & \text{if } c_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

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 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.[27]

The second case in (19) handles the situation in which the history context was never observed in training, that is, where $c_i = u_i = 0$, which would yield an undefined value

25 Collins 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, personal communication, January 2003). 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 (16) or (18) depending on the value of a particular run-time setting.

26 The 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 $\overline{n_i} = \frac{c_i}{u_i}$, the average number of transitions from the history context $\phi_i(B)$ to a possible future. With a little algebraic manipulation, we have

$$\lambda_i = \frac{\overline{n_i}}{\overline{n_i} + 5}$$

a quantity that is at its maximum when $\overline{n_i} = c_i$ and at its minimum when $\overline{n_i} = 1$, 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 (one 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 $p(\cdot \mid \phi_i(B))$.

27 As mentioned above, the P_{prior_w} parameters are unsmoothed. However, as a result of 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.

Table 1
Back-off levels for P_{L_w}/P_{R_w}, the modifier headword generation parameter classes. $w_{L_i}$and t_{L_i} are, respectively, the headword 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.

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

Table 2
Our new parameter class for the generation of headwords of modifying nonterminals.

Back-off level	$P_{M_w}(w_{M_i} \mid \ldots)$
0	$\gamma(M_i), t_{M_i}, \alpha(P), \gamma(H), w_h, t_h, subcat_{side}, \texttt{vi}(M_i), \delta(M_{i-1}), side$
1	$\gamma(M_i), t_{M_i}, \alpha(P), \gamma(H), t_h, subcat_{side}, \texttt{vi}(M_i), \delta(M_{i-1}), side$
2	t_{M_i}

when $f_t = 0$. In such situations, making $\lambda_i = 0$ throws all remaining probability mass 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, $\tilde{e}_i = \tilde{e}_{i+1}$.[28]

6.9 Modifier Head-Word Generation

As mentioned in Section 6.4, 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 headword 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 are specified in Table 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 headword of a modifying nonterminal is conditioned only on its part of speech.

6.9.1 Smoothing and the Last Level of Back-Off. Table 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 headword, resulting in a general estimate $\hat{p}(w \mid t)$, as opposed to $\hat{p}_{L_w}(w_{L_i} \mid t_{L_i})$. Accordingly, in our emulation of Collins' model, we replace the left- and right-word parameter classes with a single modifier headword generation parameter class that, as with (11), includes a boolean *side* component that is deleted from the last level of back-off (see Table 2).

Even with this change, there is still a problem. Every headword in a lexicalized parse tree is the modifier of some other headword—*except* the word that is the head of the entire sentence (i.e., the headword of the root nonterminal). In order to properly duplicate Collins' model, an implementation must take care that the $P(w \mid t)$ model includes counts for these important headwords.[29]

28 This fact is crucial in understanding how little the Collins parsing model relies on bilexical statistics, as described in Section 8.2 and the supporting experiment shown in Table 6.

29 In our implementation, we add such counts by having our trainer generate a "fake" modifier event in

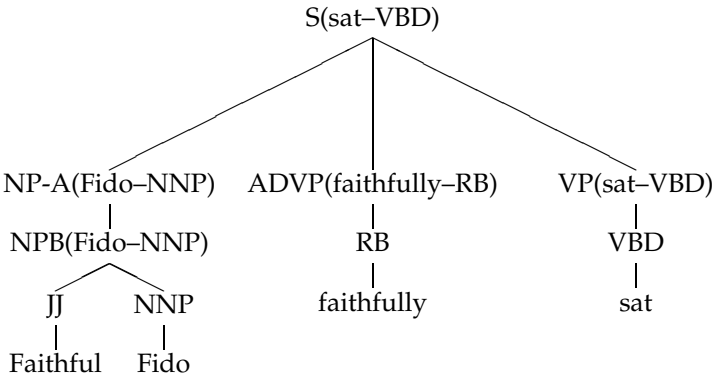


Figure 11
The low-frequency word *Fido* is mapped to +UNKNOWN+, but only when it is generated, not when it is conditioned upon. All the nonterminals have been lexicalized (except for preterminals) to show where the heads are.

6.9.2 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 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 maps only 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 11, where we assume *Fido* is a low-frequency word, the trainer would derive counts for the smoothed parameter

$$p_{L_w}\left(+\texttt{UNKNOWN}+ \mid \texttt{NP-A}, \texttt{NNP}, \mathit{coord} = 0, \mathit{punc} = 0, \texttt{S}, \texttt{VP}, \texttt{sat}, \texttt{VBD}, \ldots\right) \tag{20}$$

However, when collecting events that condition on *Fido*, such as the parameters

$$p_L\left(\texttt{JJ(JJ)} \mid \texttt{NPB}, \texttt{NNP}, \texttt{Fido}\right)$$

$$p_{L_w}\left(\texttt{Faithful} \mid \texttt{JJ}, \texttt{JJ}, \texttt{NPB}, \texttt{NNP}, \texttt{Fido}\right)$$

the word would not be mapped.

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 such a truly unknown word in the history, it will necessarily throw all probability mass to the backed-off estimate ($\tilde{e}_1$ 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"[30] model, if all other parts of the model are probabilistically sound (which is actually

which 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 Section 6.10 for details on the +TOP+ nonterminal).

30 The 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.

Table 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. n/a: not applicable.

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

not the case here). With a parsing model such as Collins' that uses bilexical dependencies, generating words in the course of parsing is done very much as it is in 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 those in a statistical parser that uses bilexical dependencies derived from head-lexicalization.

Collins (personal communication, January 2003) notes that his parser's unknown-word-mapping scheme could be made consistent if one were to add a parameter class that estimated $\hat{p}(w \mid$ +UNKNOWN+$)$, where $w \in V_L \cup \{$+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 $\arg\max_T P(T \mid S)$.

6.10 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 generated observed roots, one for generating the partially lexicalized root nonterminal, which we call $P_{TOP_{NT}}$, and the other for generating the headword of the entire sentence, which we call P_{TOP_w}. Table 3 gives the unpublished back-off structure of these two additional parameter classes.

Note that P_{TOP_w} backs off to simply estimating $\hat{p}(w \mid t)$. Technically, it should be estimating $\hat{p}_{NT}(w \mid t)$, which is to say the probability of a word's occurring with a tag in the space of lexicalized nonterminals. This is different from the last level of back-off in the modifier headword parameter classes, which is effectively estimating $\hat{p}(w \mid t)$ in the space of lexicalized preterminals. The difference is that in the same sentence, the same headword can occur with the same tag in multiple nodes, such as *sat* in Figure 8, which occurs with the tag VBD three times (instead of just once) in the tree shown there. Despite this difference, Collins' parser uses counts from the (shared) last level of back-off of the P_{L_w} and 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.

7. 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 pseu-

docode version of his algorithm in an appendix. This section contains a few practical details.

7.1 Chart Item Equality

Since the goal of the decoding process is to determine the maximally likely theory, if during decoding a proposed chart item is equal (or, technically, equivalent) to an item that is already in the chart, the one with the greater score survives. Chart item equality is closely tied to the generative parameters used to construct theories: We want to treat two chart items as unequal if they represent derivation forests that would be considered unequal 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 equal, they must have the same label (the label of the root of their respective derivation forests' subtrees), the same headword 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.

7.2 Pruning

Ideally, every parse theory could be kept in the chart, and when the root symbol has been generated for all theories, 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 theory 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 Table 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 after he added the base NP model, to handle the greater perplexity associated with NPs.

The second form of pruning employed is 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 theories that would generate constituents that violate this comma constraint.[31] 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

31 If 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.

Table 4
Overall parsing results using only details found in Collins (1997, 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 (labeled recall) and LP (labeled precision) are the primary scoring metrics. CBs is the number of crossing brackets. 0 CBs and ≤ 2 CBs are the percentages of sentences with 0 and ≤ 2 crossing brackets, respectively. F (the F-measure) is the evenly weighted harmonic mean of precision and recall, or $\frac{LP \cdot LR}{\frac{1}{2}(LP+LR)}$.

	Performance on Section 00					
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**
Clean-room Model 2	88.85	88.97	0.92	65.55	87.06	**88.91**

sentence occurs after an opening 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 theories of an NPB subtree. As it turns out, using the comma constraint also affects accuracy, as shown in Section 8.1.

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 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 particular 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.[32]

7.3 Unknown Words and Parts of Speech

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

8. Evaluation

8.1 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, 1999), as shown in Table 4.

The clean-room model has a 10.6% increase in F-measure error 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

32 Although 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.

Table 5
Effects of independently removing or 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 Section 6.10), and $p(w \mid t)$ estimates were side-specific (see Section 6.9.1). See Table 4 for definitions of column headings.

	Performance on Section 00					
Model description	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 Section 5.3)	89.94	90.22	0.77	68.99	89.27	90.08
No training trees skipped (see Section 5.2)	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
Nondeficient estimation (see Section 6.8.1)	89.75	90.00	0.80	68.82	88.88	89.87
No comma constraint (see Section 7.2)	89.52	89.80	0.84	68.09	88.20	89.66
No universal $p(w \mid t)$ 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

error seen when removing such published features as the verb-intervening component of the distance metric, which results in an F-measure error increase of 9.86%, or the subcat feature, which results in a 7.62% increase in F-measure error.[33] Therefore, while the collection of unpublished details presented in Sections 4–7 is disparate, in toto those details are every bit as important to overall parsing performance as certain of the published features.

This does not mean that all the details are equally important. Table 5 shows the effect on overall parsing performance of independently removing or changing certain of the more than 30 unpublished details.[34] Often, the detrimental effect of a particular change is quite insignificant, even by the standards of the performance-obsessed world of statistical parsing, and occasionally, the effect of a change is not even detrimental at all. That is why we do not claim the 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 $p(w \mid t)$ model, *do* appear to have a much more marked effect on overall parsing accuracy than others.

8.2 Bilexical Dependencies

The previous section 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 subcats (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

33 These F-measures and the differences between them were calculated from experiments presented in Collins (1999, page 201); these experiments, unlike those on which our reported numbers are based, 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.

34 As a reviewer 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 Section 7.2.

Table 6
Number of times our parsing engine was able to deliver a probability for the various levels of back-off of the modifier-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.

Back-off level	Number of accesses	Percentage
0	3,257,309	1.5
1	24,294,084	11.0
2	191,527,387	87.4
Total	219,078,780	100.0

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) indicates 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) reimplemented Collins' Model 1 (essentially Model 2 but without subcats) and altered the P_{L_w} and P_{R_w} parameters so that they no longer had the top level of context that included the headword (he removed back-off level 0, as depicted in Table 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 able to reimplement Collins' baseline model only partially, and the performance of his partial reimplementation was not quite as good as that of Collins' parser.[35]

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 obtains a recall of 89.89% and a precision of 90.14% on sentences of length ≤ 40 words (see Table 8, Model $\mathcal{M}_{tw,tw}$). Without bigrams, performance drops only to 89.49% on recall, 89.95% on precision—an exceedingly small drop in performance (see Table 8, Model $\mathcal{M}_{tw,t}$). In an additional experiment, we have examined the number of times that the parser is able, while decoding Section 00, to deliver a requested probability for the modifier-word generation model using the increasingly less-specific contexts of the three back-off levels. The results are presented in Table 6. Back-off level 0 indicates the use of the full history context, which contains the head-child's headword. Note that probabilities making use of this full context, that is, making use of bilexical dependencies, are available only 1.49% of the time. Combined with the results from the previous experiment, this suggests rather convincingly that such statistics are far less significant than once thought to the overall discriminative power of Collins' models, confirming Gildea's result for Model 2.[36]

35 The reimplementation was necessarily only partial, as Gildea did not have access to all the unpublished details of Collins' models that are presented in this article.

36 On 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 1,917 sentences of Section 00. Among other things, this

Table 7
Results on Section 00 with simplified head rules. The baseline model is our engine in its closest possible emulation of Collins' Model 2. See Table 4 for definitions of column headings.

LR	LP	CBs	0 CBs	$\leq$ 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

8.3 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 Chiang and Bikel (2002), we explored methods for recovering latent information in treebanks. The second half of that paper focused on a use of the Inside–Outside algorithm to reestimate the parameters of a model defined over an **augmented tree space**, where the observed data were considered to be the gold-standard labeled bracketings found in the treebank, and the hidden data were considered to be the head-lexicalizations, one of the most notable tree augmentations performed by modern statistical parsers. These expectation maximization (EM) experiments were motivated by the desire to overcome the limitations imposed by the heuristics that have been heretofore used to perform head-lexicalization in treebanks. In particular, it appeared that the head rules used in Collins' parser had been tweaked specifically for the English Penn Treebank. Using EM would mean 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 in which 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 $\leq$40 words in Section 00 (see Table 7). So contrary to our expectations, the lack of careful head-choice is not crippling in allowing the parser to disambiguate competing theories and is a further indication that semantic preferences, as represented by conditioning on a headword, rarely override structural ones.

8.4 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, that is, structures being generated conditioning on headwords and headwords being generated conditioning on structures.

To test the effect of such lexicostructural 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 re-

certainly points to the utility of caching probabilities (the 219 million are tokens, not types).

Table 8
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 headword 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 headword 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}. See Table 4 for definitions of the remaining column headings.

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
	$\mathcal{M}_{tw,\phi}$	✓	✓			89.24	89.86	0.81	66.80	88.76	89.55
	$\mathcal{M}_{t,\phi}$	✓				88.01	88.96	0.91	63.93	86.91	88.48
	$\mathcal{M}_{\phi,\phi}$					87.01	88.75	0.96	61.08	86.00	87.87

sults are shown in Table 8. Model $\mathcal{M}_{tw,tw}$ shows our baseline, and Model $\mathcal{M}_{\phi,\phi}$ shows the effect of removing *all* dependence on the headword and its part of speech, with the other models illustrating varying degrees of removing elements from the two parameter classes' conditioning contexts. Notably, including the headword w_h in or removing it 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 $\mathcal{M}_{tw,\phi}$ to Model $\mathcal{M}_{t,\phi}$. This reinforces the notion that particular headwords have structural preferences, so that making the P_M parameters dependent on headwords would capture such preferences. As for effects involving dependence on the head tag t_h, observe that moving from Model $\mathcal{M}_{tw,t}$ to Model $\mathcal{M}_{tw,\phi}$ results in a small drop in both recall and precision, whereas making an analogous move from Model $\mathcal{M}_{t,t}$ to Model $\mathcal{M}_{t,\phi}$ results in 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 headwords matter far more than parts of speech for determining structural preferences, as one would expect.

9. Conclusion

We have documented what we believe is the complete set of heretofore unpublished details Collins used in his parser, such that, along with Collins' (1999) thesis, thi s article contains all information necessary to duplicate Collins' benchmark results. Indeed, these as-yet-unpublished details account for an 11% relative increase in error from an implementation including all details to a clean-room implementation of Collins' model. We have also shown a cleaner and equally well-performing method for the handling of punctuation and conjunction, and we have revealed certain other probabilistic oddities about Collins' parser. We have not only analyzed the effect of the unpublished details but also reanalyzed the effect of certain well-known details, revealing that bilexical dependencies are barely used by the model and that head choice is not nearly as important to overall parsing performance as once thought. Finally, we have performed experiments that show that the true discriminative power of lexicalization appears to lie in the fact that unlexicalized syntactic structures are generated conditioning on the headword and head tag. These results regarding the

lack of reliance on bilexical statistics suggest that generative models still have room for improvement through the employment of bilexical-class statistics, that is, dependencies among head-modifier word *classes*, where such classes may be defined by, say, WordNet synsets. Such dependencies might finally be able to capture the semantic preferences that were thought to be captured by standard bilexical statistics, as well as to alleviate the sparse-data problems associated with standard bilexical statistics. This is the subject of our current research.

Appendix: Complete List of Parameter Classes

This section 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 this article. 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 alpha and gamma are defined in Section 6.1. 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	$P_H(H\|\ldots)$	$P_G(G\|\ldots)$ $P_{subcat_L}(subcat_L\|\ldots)$ $P_{subcat_R}(subcat_R\|\ldots)$
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 headword of the entire sentence).

Back-off level	$P_M\left(M(t)_i, \texttt{coord}, \texttt{punc}\,\middle\|\ldots\right)$
0	$\alpha(P)$, $\gamma(H)$, w_h, t_h, Δ_{side}, $subcat_{side}$, $side$
1	$\alpha(P)$, $\gamma(H)$, t_h, Δ_{side}, $subcat_{side}$, $side$
2	$\alpha(P)$, $\gamma(H)$, Δ_{side}, $subcat_{side}$, $side$

Back-off level	$P_{M_w}\left(w_{M_i}\middle\|\ldots\right)$
0	$\gamma(M(t)_i)$, coord, punc, $\alpha(P)$, $\gamma(H)$, w_h, t_h, Δ_{side}, $subcat_{side}$, $side$
1	$\gamma(M(t)_i)$, coord, punc, $\alpha(P)$, $\gamma(H)$, t_h, Δ_{side}, $subcat_{side}$, $side$
2	t_{M_i}

The two parameter classes for generating modifying nonterminals that are children of base NPs (NPB nodes), $P_{M,\texttt{NPB}}$ and $P_{M_w,\texttt{NPB}}$, have the following back-off structures. Back-off level 2 of the $P_{M_w,\texttt{NPB}}$ parameters includes words that are the heads of the observed roots of sentences (that is, the headword of the entire sentence). Also, note that there is no coord flag, as coordinating conjunctions are generated in the same way as 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	$P_{M,\texttt{NPB}}\left(M(t)_i, \texttt{punc} \mid \ldots\right)$	$P_{M_w,\texttt{NPB}}\left(w_{M_i} \mid \ldots\right)$
0	P, $M(w,t)_{i-1}$, *side*	M_i, t_{M_i}, punc, P, $M(w,t)_{i-1}$, *side*
1	P, $M(t)_{i-1}$, *side*	M_i, t_{M_i}, punc, P, $M(t)_{i-1}$, *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, personal communication, October 2001), 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 or t_c is the particular preterminal (part-of-speech tag) that is conjoining the modifier to the head-child (such as CC or :);
- w_p or w_c is the particular word that is conjoining the modifier to the head-child (such as and or :).

Back-off level	$P_{coord}(t_c \mid \ldots)$ $P_{punc}(t_p \mid \ldots)$	$P_{coord_w}(w_c \mid \ldots)$ $P_{punc_w}(w_p \mid \ldots)$
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 Table 3; n/a: not applicable).

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

The parameter classes for generating prior 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 note 27).

Back-off level	$P_{prior_w}(w, t \mid \ldots)$	$P_{prior_{NT}}(M \mid \ldots)$
0	prior	w, t
1	prior	t

Acknowledgments

I would especially like to thank Mike Collins for his invaluable assistance and great generosity while I was replicating his thesis results and for his comments on a prerelease draft of this article. Many thanks to David Chiang and Dan Gildea for the many valuable discussions during the course of this work. Also, thanks to the anonymous reviewers for their helpful and astute observations. Finally, thanks to my Ph.D. advisor Mitch Marcus, who during the course of this work was, as ever, a source of keen insight and unbridled optimism. This work was supported in part by NSF grant no. SBR-89-20239 and DARPA grant no. N66001-00-1-8915.

References

Baker, J. K. 1979. Trainable grammars for speech recognition. In *Spring Conference of the Acoustical Society of America*, pages 547–550, Boston.

Bies, A. 1995. *Bracketing guidelines for Treebank II style Penn Treebank Project*. Available at 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.

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, DC.

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 Fifth DARPA Speech and Natural Language Workshop*, Harriman, NY.

Booth, T. L. and R. A. Thompson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, volume C-22: 442–450.

Chiang, David and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of COLING'02*, Taipei.

Cohen, Paul R. 1995. *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge, MA.

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, Santa Cruz, CA.

Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL-EACL '97*, pages 16–23, Madrid.

Collins, Michael. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Collins, Michael. 2000. Discriminative reranking for natural language parsing. In *International Conference on Machine Learning*, Stanford University, Stanford, CA.

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.

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.

Gildea, Daniel. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, Pittsburgh.

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.

Goodman, Joshua. 1997. Global thresholding and multiple-pass parsing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Brown University, Providence, RI.

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, MD.

Hwa, Rebecca. 2001. On minimizing training corpus for parser acquisition. In *Proceedings of the Fifth Computational Natural Language Learning Workshop*, Toulouse, France, 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, June.

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.

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.

Ratnaparkhi, Adwait. 1996. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing*, University of Pennsylvania, Philadelphia, May.

Witten, I. T. and T. C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory* 37: 1085–1094.