Head-Driven Statistical Models for Natural Language Parsing

Michael Collins*
MIT Artificial Intelligence Laboratory

This paper describes three statistical models for natural language parsing. The models extend methods from Probabilistic Context Free Grammars to lexicalized grammars, leading to approaches where a parse tree is represented as the sequence of decisions corresponding to a head-centered, top-down derivation of the tree. Independence assumptions then lead to parameters that encode the X-bar schema, subcategorization, ordering of complements, placement of adjuncts, bigram lexical dependencies, wh-movement, and preferences for close attachment. All of these preferences are expressed by probabilities conditioned on lexical heads. The models are evaluated on the Penn Wall Street Journal treebank, showing that their accuracy is competitive with other models in the literature. In order to gain a better understanding of the models, we also give results on different constituent types, and give a breakdown of precision/recall results in recovering various types of dependencies. We analyse various characteristics of the models through experiments on parsing accuracy, by collecting frequencies of various structures in the treebank, and through linguistically motivated examples. Finally, we compare the models to others that have been applied to parsing the treebank, aiming to give some explanation of the difference in performance of the various models.

1 Introduction

Ambiguity is a central problem in natural language parsing. Combinatorial effects mean that even relatively short sentences can receive a considerable number of parses under a wide-coverage grammar. Statistical parsing approaches tackle the ambiguity problem by assigning a probability to each parse tree, thereby ranking competing trees in order of plausibility. In many statistical models the probability for each candidate tree is calculated as a product of terms, each term corresponding to some sub-structure within the tree. The choice of *parameterization* is essentially the choice of how to represent parse trees. There are two critical questions regarding the parameterization of the problem:

1. Which linguistic objects (e.g., context-free rules, parse moves) should the model's parameters be associated with? In other words, which features should be used to discriminate between alternative parse trees?

2. How can this choice be instantiated in a sound probabilistic model?

In this paper we explore these issues within the framework of generative models, more precisely the history-based models originally introduced to parsing by (Black et al. 1992b). In a history-based model a parse tree is represented as a sequence of decisions,

* Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, USA

the decisions being made in some derivation of the tree. Each decision has an associated probability, and the product of these probabilities defines a probability distribution over possible derivations.

We first describe three parsing models based on this approach. The models were originally introduced in (Collins 1997); the current paper[1] gives considerably more detail and discussion of the models. In **Model 1** we show one approach that extends methods from Probabilistic Context Free Grammars (PCFGs) to lexicalized grammars. Most importantly, the model has parameters corresponding to dependencies between pairs of head-words. We also show how to incorporate a "distance" measure into these models, by generalizing the model to a history-based approach. The distance measure allows the model to learn a preference for close-attachment, or right-branching structures.

In **Model 2**, we extend the parser to make the complement/adjunct distinction, which will be important for most applications using the output from the parser. Model 2 is also extended to have parameters corresponding directly to probability distributions over subcategorization frames for head-words. The new parameters lead to an improvement in accuracy.

In **Model 3** we give a probabilistic treatment of wh-movement, which is loosely based on the analysis of wh-movement in Generalized Phrase Structure Grammar (Gazdar et al. 1995). The output of the parser is now enhanced to show trace co-indexations in wh-movement cases. The parameters in this model are interesting in that they correspond directly to the probability of propagating GPSG-style *slash* features through parse trees, potentially allowing the model to learn island constraints.

In the resulting models a parse tree is represented as the sequence of decisions corresponding to a head-centered, top-down derivation of the tree. Independence assumptions then follow naturally, leading to parameters that encode the X-bar schema, subcategorization, ordering of complements, placement of adjuncts, lexical dependencies, wh-movement, and preferences for close attachment. All of these preferences are expressed by probabilities conditioned on lexical heads. For this reason we refer to the models as *head-driven statistical models*.

We describe evaluation of the three models on the Penn Wall Street Journal treebank (Marcus et al. 1993). Model 1 achieves 87.7/87.5% constituent precision and recall on sentences of up to 100 words in length in section 23 of the treebank, and Models 2 and 3 give further improvements to 88.3/88.0% constituent precision/recall. These results are competitive with other models which have been applied to parsing the Penn treebank. Models 2 and 3 produce trees with information about wh-movement or subcategorization. Many NLP applications will need this information to extract predicate-argument structure from parse trees.

The rest of the paper is structured as follows. Section 2 gives background material on probabilistic context-free grammars, and describes how rules can be "lexicalized" through the addition of head-words to parse trees. Section 3 introduces the three probabilistic models. Section 4 describes various refinments to these models. Section 5 discusses issues of parameter estimation, the treatment of unknown words, and also the parsing algorithm. Section 6 gives results evaluating the performance of the models on the Penn Wall Street Journal treebank (Marcus et al. 1993). Section 7 investigates various aspects of the models in more detail. We give a detailed analysis of the parser's performance on treebank data, including results on different constituent types. We also give a breakdown of precision/recall results in recovering various types of dependencies. The intention is to give a better idea of the strengths and weaknesses of the parsing models. Section 7

1 Much of this paper is an edited version of chapters 7 and 8 of (Collins 1999).

goes on to discuss the distance features in the models, the implicit assumptions that the models make about the treebank annotation style, and finally discusses the way that context-free rules in the original treebank are broken-down, allowing the models to generalize by producing new rules on test-data examples. We analyse these phenomena through experiments on parsing accuracy, by collecting frequencies of various structures in the treebank, and through linguistically motivated examples. Finally, section 8 gives more discussion, by comparing the models to others that have been applied to parsing the treebank. We aim to give some explanation of the difference in performance of the various models.

2 Background

2.1 Probabilistic Context-Free Grammars

Probabilistic context-free grammars are the starting point for the models in this paper. For this reason we briefly recap the theory behind non-lexicalized PCFGs, before moving to the lexicalized case.

Following (Hopcroft and Ullman 79), a context-free grammar G is a 4-tuple (N, Σ, A, R) where N is a set of non-terminal symbols, Σ is an alphabet, A is a distinguished start symbol in N, and R is a finite set of rules, where each rule is of the form $X \rightarrow \beta$ for some $X \in N$, $\beta \in (N \cup \Sigma)^*$. The grammar defines a set of possible strings in the language, and also defines a set of possible left-most derivations under the grammar. Each derivation corresponds to a tree-sentence pair that is well formed under the grammar.

A probabilistic context-free grammar is a simple modification of a context-free grammar, where each rule in the grammar has an associated probability $P(\beta \mid X)$. This can be interpreted as the conditional probability of X being expanded using the rule $X \rightarrow \beta$, as opposed to one of the other possibilities for expanding X listed in the grammar. The probability of a derivation is then a product of terms, each term corresponding to a rule application in the derivation. For a tree-sentence pair (T, S) derived by n applications of context-free rules $\mathrm{LHS}_i \rightarrow \mathrm{RHS}_i$, $1 \leq i \leq n$, its probability under the PCFG is

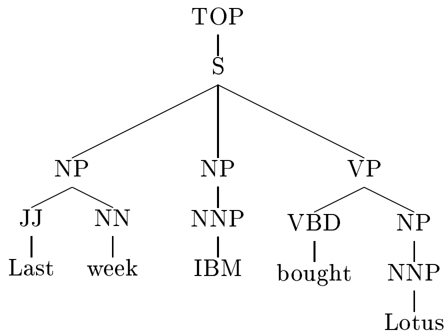
$$P(T, S) = \prod_{i=1}^{n} P(\mathrm{RHS}_i \mid \mathrm{LHS}_i)$$

(LHS stands for "Left hand side", RHS stands for "Right hand side"). (Booth and Thompson 73) give conditions under which the PCFG does in fact define a distribution over the possible derivations (trees) generated by the underlying grammar. The first condition is that the rule probabilities define conditional distributions over how each non-terminal in the grammar can expand. The second is a technical condition that guarantees that the stochastic process generating trees terminates in a finite number of steps with probability 1.

A central problem in PCFGs is to define the conditional probability $P(\beta \mid X)$ for each rule $X \rightarrow \beta$ in the grammar. A simple way to do this is to take counts from a treebank and then to use the maximum likelihood estimate

$$P(\beta \mid X) = \frac{Count(X \rightarrow \beta)}{Count(X)} \tag{1}$$

If the treebank has actually been generated from a probabilistic context-free grammar with the same rules and non-terminals as the model, then in the limit as the training sample size approaches infinity the probability distribution implied by these estimates



Internal rules			Lexical rules		
TOP	$\rightarrow$	S	JJ	$\rightarrow$	Last
S	$\rightarrow$	NP NP VP	NN	$\rightarrow$	week
NP	$\rightarrow$	JJ NN	NNP	$\rightarrow$	IBM
NP	$\rightarrow$	NNP	VBD	$\rightarrow$	bought
VP	$\rightarrow$	VBD NP	NNP	$\rightarrow$	Lotus
NP	$\rightarrow$	NNP			

Figure 1
A non-lexicalized parse tree, and a list of the rules it contains.

will converge to the distribution of the underlying grammar.[2]

Once the model has been trained, we have a model that defines $P(T, S)$ for any sentence-tree pair in the grammar. The output on a new test sentence S is the most likely tree under this model,

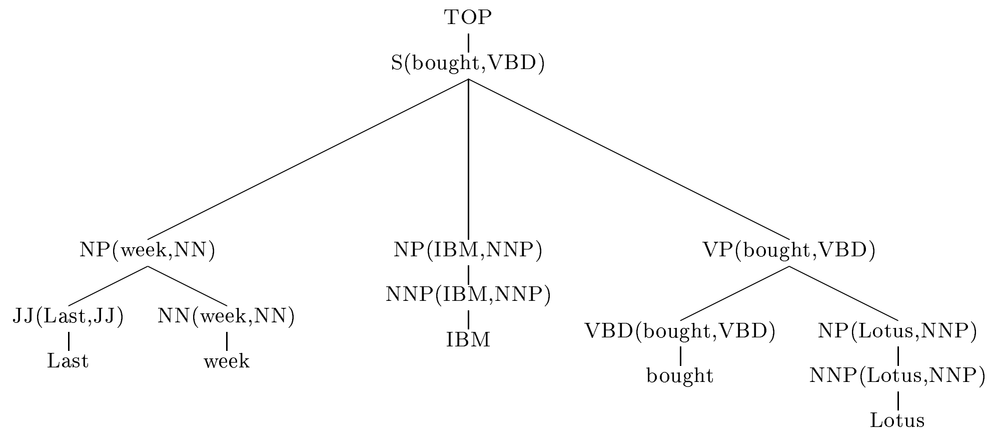
$$T_{best} = \arg\max_T P(T \mid S) = \arg\max_T \frac{P(T,S)}{P(S)} = \arg\max_T P(T,S)$$

The parser itself is an algorithm that searches for the tree, T_{best}, that maximises $P(T, S)$. In the case of PCFGs this can be accomplished using a variant of the CKY algorithm applied to weighted grammars (providing that the PCFG can be converted to an equivalent PCFG in Chomsky Normal Form) – see for example (Manning and Schütze 1999).

If the model probabilities $P(T, S)$ are the same as the true distribution generating training and test examples, returning the most likely tree under $P(T, S)$ will be optimal in terms of minimizing the expected error rate (number of incorrect trees) on newly drawn test examples. Hence if the data is generated by a PCFG, and there are enough training examples for the maximum-likelihood estimates to converge to the true values, then this parsing method will be optimal. In practice, these assumptions can not be verified, and are arguably quite strong, but these limitations have not prevented generative models from being successfully applied to many NLP and speech tasks. See (Collins 2002) for a discussion of other ways of conceptualizing the parsing problem.

In the Penn treebank (Marcus et al. 1993), that is the source of data for our experiments, the rules are either internal to the tree, where LHS is a non-terminal and RHS is a string of one or more non-terminals, or lexical, where LHS is a part-of-speech tag and

2 This point is actually more subtle than it first appears – we thank one of the anonymous reviewers for pointing this out – and we were unable to find proofs of this property in the literature for PCFGs. The *rule* probabilities for any non-terminal which appears with probability > 0 in parse derivations will converge to their underlying values, by the usual properties of maximum-likelihood estimation for multinomial distributions. Assuming that the underlying PCFG generating training examples meets both criteria in (Booth and Thompson 73), it can be shown that convergence of rule probabilities implies that the distribution over *trees* will converge to that of the underlying PCFG, at least when Kullback-Liebler divergence or the infinity-norm is taken to be the measure of distance between the two distributions. Thanks to Tommi Jaakkola and Nathan Srebro for discussions on this topic.



Internal Rules:

`TOP`	→	`S(bought,VBD)`		
`S(bought,VBD)`	→	`NP(week,NN)`	`NP(IBM,NNP)`	`VP(bought,VBD)`
`NP(week,NN)`	→	`JJ(Last,JJ)`	`NN(week,NN)`	
`NP(IBM,NNP)`	→	`NNP(IBM,NNP)`		
`VP(bought,VBD)`	→	`VBD(bought,VBD)`	`NP(Lotus,NNP)`	
`NP(Lotus,NNP)`	→	`NNP(Lotus,NNP)`		

Lexical Rules:

`JJ(Last,JJ)`	→	`Last`
`NN(week,NN)`	→	`week`
`NNP(IBM,NNP)`	→	`IBM`
`VBD(bought,VBD)`	→	`bought`
`NNP(Lotus,NN)`	→	`Lotus`

Figure 2
A lexicalized parse tree, and a list of the rules it contains.

RHS is a word. See figure 1 for an example.

2.2 Lexicalized PCFGs

A PCFG can be lexicalized[3] by associating a word w and a part-of-speech (POS) tag t with each non-terminal X in the tree. See figure 2 for an example tree.

The PCFG model can be applied to these lexicalized rules and trees in exactly the same way as before. Whereas before the non-terminals were simple, for example `S` or `NP`, they are now extended to include a word and part-of-speech tag, for example `S(bought,VBD)` or `NP(IBM,NNP)`. Thus we write a non-terminal as $X(x)$, where $x = \langle w, t \rangle$, and X is a constituent label. Formally, nothing has changed, we have just vastly increased the number of non-terminals in the grammar (by up to a factor of $|\mathcal{V}| \times |\mathcal{T}|$, where $|\mathcal{V}|$ is the number of words in the vocabulary, and $|\mathcal{T}|$ is the number of part of speech tags).

While nothing has changed from a formal point of view, the practical consequences of expanding the number of non-terminals quickly become apparent when attempting to define a method for parameter estimation. The simplest solution would be to use the maximum-likelihood estimate as in equation 1, for example estimating the probability

3 We find lexical heads in Penn treebank data using the rules described in Appendix A of (Collins 1999). The rules are a modified version of a head table provided by David Magerman, and used in the parser described in (Magerman 1995).

associated with S(bought,VBD) $\rightarrow$ NP(week,NN) NP(IBM,NNP) VP(bought,VBD) as

$$P(\texttt{NP(week,NN) NP(IBM,NNP) VP(bought,VBD)} \mid \texttt{S(bought,VBD)}) = \frac{Count(\texttt{S(bought,VBD)} \rightarrow \texttt{NP(week,NN) NP(IBM,NNP) VP(bought,VBD)})}{Count(\texttt{S(bought,VBD)})}$$

But the addition of lexical items makes the statistics for this estimate very sparse: the count for the denominator is likely to be relatively low, and the number of outcomes (possible lexicalized RHSs) is huge, meaning that the numerator is very likely to be zero. Predicting the whole lexicalized rule in one go is too big a step.

One way to overcome these sparse data problems is to break down the generation of the RHS of each rule into a sequence of smaller steps, and then to make independence assumptions to reduce the number of parameters in the model. The decomposition of rules should aim to meet two criteria. First, the steps should be small enough for the parameter estimation problem to be feasible (i.e., in terms of having sufficient training data to train the model, providing that smoothing techniques are used to mitigate remaining sparse data problems). Second, the independence assumptions made should be linguistically plausible. In the next sections we describe three statistical parsing models which have an increasing degree of linguistic sophistication. Model 1 uses a decomposition where parameters corresponding to lexical dependencies are a natural result. The model also incorporates a preference for right-branching structures through conditioning on "distance" features. Model 2 extends the decomposition to include a step where subcategorization frames are chosen probabilistically. Model 3 handles wh-movement by adding parameters corresponding to *slash* categories being passed from the parent of the rule to one of its children, or being discharged as a trace.

3 Three Probabilistic Models for Parsing

3.1 Model 1

This section describes how the generation of the RHS of rule is broken down into a sequence of smaller steps in model 1. The first thing to note is that each internal rule in a lexicalized PCFG has the form[4]:

$$P(h) \rightarrow L_n(l_n) \ldots L_1(l_1) H(h) R_1(r_1) \ldots R_m(r_m) \tag{2}$$

H is the head-child of the rule, which inherits the head word/tag pair h from its parent P. $L_1(l_1)...L_n(l_n)$ and $R_1(r_1)...R_m(r_m)$ are left and right modifiers of H. Either n or m may be zero, and $n = m = 0$ for unary rules. Figure 2 shows a tree which will be used as an example throughout this paper. We will extend the left and right sequences to include a terminating STOP symbol, allowing a Markov process to model the left and right sequences. Thus $L_{n+1} = R_{m+1} =$ STOP.

For example, in S(bought,VBD) $\rightarrow$ NP(week,NN) NP(IBM,NNP) VP(bought,VBD):

$n = 2$	$m = 0$	$P =$ S
$H =$ VP	$L_1 =$ NP	$L_2 =$ NP
$L_3 =$ STOP	$R_1 =$ STOP	h = $\langle$bought, VBD$\rangle$
$l_1 = \langle$IBM, NNP$\rangle$	$l_2 = \langle$week, NN$\rangle$	

Note that lexical rules, in contrast to the internal rules, are completely deterministic. They always take the form

$$P(h) \rightarrow w$$

4 With the exception of the top rule in the tree, which has the form TOP $\rightarrow$ H(h).

where P is a part-of-speech tag, h is a word-tag pair $\langle w, t\rangle$, and the rule rewrites to just the word w. See figure 2 for examples of lexical rules. Formally, we will always take a lexicalized non-terminal $P(h)$ to expand deterministically (with probability 1) in this way if P is a part-of-speech symbol. Thus for the parsing models we require the non-terminal labels to be partitioned into two sets, part-of-speech symbols and other non-terminals. Internal rules always have an LHS where P is not a part-of-speech symbol. Because lexicalized rules are deterministic they will not be discussed in the remainder of this paper: all of the modeling choices concern internal rules.

The probability of an internal rule can be rewritten (exactly) using the chain rule of probabilities:

$$P(L_{n+1}(l_{n+1})...L_1(l_1)H(h)R_1(r_1)...R_{m+1}(r_{m+1}) \mid P,h) =$$
$$P_h(H \mid P,h) \times \prod_{i=1...n+1} P_l(L_i(l_i) \mid L_1(l_1)...L_{i-1}(l_{i-1}),P,h,H) \times$$
$$\prod_{j=1...m+1} P_r(R_j(r_j) \mid L_1(l_1)...L_{n+1}(l_{n+1}),R_1(r_1)...R_{j-1}(r_{j-1}),P,h,H)$$

(the subscripts h, l and r are used to denote the head, left-modifier and right-modifier parameter types respectively).

Next, we make the assumption that the modifiers are generated independently of each other,

$$P_l(L_i(l_i) \mid L_1(l_1)...L_{i-1}(l_{i-1}),P,h,H) \quad = \quad P_l(L_i(l_i) \mid P,h,H) \tag{3}$$

$$P_r(R_j(r_j) \mid L_1(l_1)...L_{n+1}(l_{n+1}),R_1(r_1)...R_{j-1}(r_{j-1}),P,h,H) \quad = \quad P_r(R_j(r_j) \mid P,h,H) \tag{4}$$

In summary, the generation of the RHS of a rule such as (2), given the LHS, has been decomposed into three steps[5]:

1. Generate the head constituent label of the phrase, with probability $P_h(H \mid P,h)$.

2. Generate modifiers to the left of the head with probability $\prod_{i=1..n+1} P_l(L_i(l_i) \mid P,h,H)$, where $L_{n+1}(l_{n+1}) = \texttt{STOP}$. The `STOP` symbol is added to the vocabulary of non-terminals, and the model stops generating left modifiers when it is generated.

3. Generate modifiers to the right of the head with probability $\prod_{i=1..m+1} P_r(R_i(r_i) \mid P,h,H)$. We define $R_{m+1}(r_{m+1})$ as `STOP`.

For example, the probability of the rule `S(bought)` $\rightarrow$ `NP(week) NP(IBM) VP(bought)` would be estimated as

$$P_h(\texttt{VP} \mid \texttt{S,bought}) \times P_l(\texttt{NP(IBM)} \mid \texttt{S,VP,bought}) \times P_l(\texttt{NP(week)} \mid \texttt{S,VP,bought})$$
$$\times P_l(\texttt{STOP} \mid \texttt{S,VP,bought}) \times P_r(\texttt{STOP} \mid \texttt{S,VP,bought})$$

In this example, and in the examples in the rest of the paper, for brevity we omit the part-of-speech tags associated with words, for example writing `S(bought)` rather than `S(bought,VBD)`. We emphasize that throughout the models in this paper, each word is always paired with its part of speech: either when the word is generated, or when the word is being conditioned upon.

5 An exception is the first rule in the tree, `TOP` $\rightarrow$ `H(h)`, which has probability $P_{TOP}(H,h|TOP)$

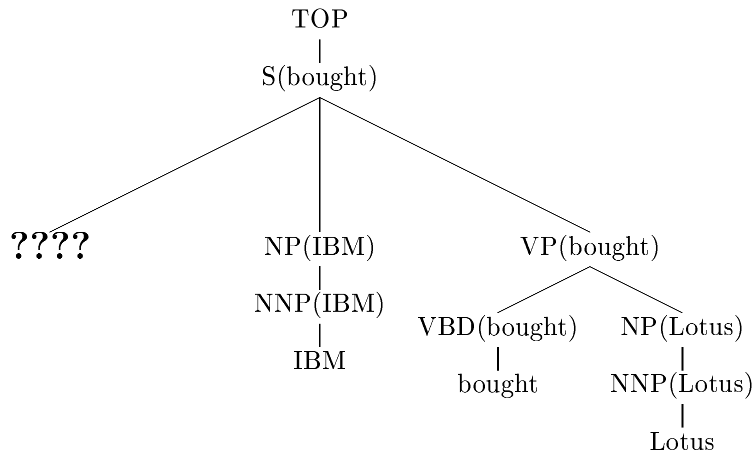


Figure 3
A partially completed tree derived depth-first. ???? marks the position of the next modifier to be generated — it could be a non-terminal/head-word/head-tag triple, or the `STOP` symbol. The distribution over possible symbols in this position could be conditioned on any previously generated structure, i.e., any structure appearing in the figure.

3.1.1 Adding Distance to the model In this section we first describe how the model can be extended to be "history-based". We then show how this extension can be utilized in incorporating "distance" features in the model.

(Black et al. 1992b) originally introduced history-based models for parsing. Equations 3 and 4 made the independence assumption that each modifier is generated independently of the others (i.e., that the modifiers are generated independently of everything except P, H and h). In general, however, the probability of generating each modifier could depend on any function of the previous modifiers, head/parent category and head word. Moreover, if the top down derivation order is fully specified, then the probability of generating a modifier can be conditioned on any of the structure that has been previously generated. The remainder of this paper assumes that the derivation order is depth-first — that is, each modifier recursively generates the sub-tree below it before the next modifier is generated. Figure 3 gives an example that illustrates this.

The models in (Collins 1996) showed that the distance between words standing in head-modifier relationships was important, in particular that it is important to capture a preference for right-branching structures (which almost translates into a preference for dependencies between adjacent words), and a preference for dependencies not to cross a verb. In this section we describe how this information can be incorporated into model 1. In section 7.2 of this paper we describe experiments that evaluate the effect of these features on parsing accuracy.

Distance can be incorporated into the model by modifying the independence assumptions so that each modifier has a limited dependence on the previous modifiers:

$$P_l(L_i(l_i) \mid H, P, h, L_1(l_1)...L_{i-1}(l_{i-1})) = P_l(L_i(l_i) \mid H, P, h, distance_l(i-1)) \tag{5}$$

$$P_r(R_i(r_i) \mid H, P, h, R_1(r_1)...R_{i-1}(r_{i-1})) = P_r(R_i(r_i) \mid H, P, h, distance_r(i-1)) \tag{6}$$

Here $distance_l$ and $distance_r$ are functions of the surface string below the previous modifiers. See figure 4 for illustration. The distance measure is similar to that in (Collins 1996), a vector with the following 2 elements: (1) is the string of zero length? (2) does

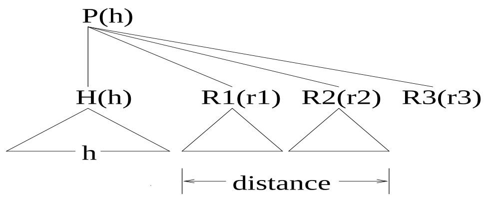


Figure 4
The next child, $R_3(r_3)$, is generated with probability $P(R_3(r_3) \mid P, H, h, distance_r(2))$. The *distance* is a function of the surface string below previous modifiers R_1 and R_2. In principle the model could condition on any structure dominated by H, R_1 or R_2 (or, for that matter, on any structure previously generated elsewhere in the tree).

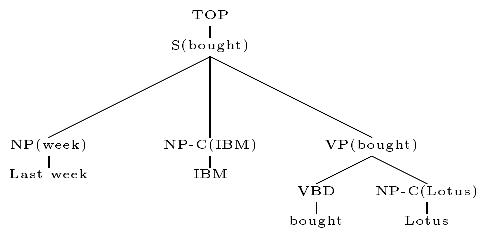


Figure 5
A tree with the `-C` suffix used to identify complements. "IBM" and "Lotus" are in subject and object position respectively. "Last week" is an adjunct.

the string contain a verb? The first feature allows the model to learn a preference for right-branching structures. The second feature[6] allows the model to learn a preference for modification of the most recent verb.[7]

3.2 Model 2: The complement/adjunct distinction and subcategorization

The tree in figure 2 illustrates the importance of the complement/adjunct distinction. It would be useful to identify "IBM" as a subject, and "Last week" as an adjunct (temporal modifier), but this distinction is not made in the tree, as both `NP`s are in the same position[8] (sisters to a `VP` under an `S` node). From here on we will identify complements[9] by attaching a `-C` suffix to non-terminals. Figure 5 shows the tree in figure 2 with added complement markings.

A post-processing stage could add this detail to the parser output, but there are a couple of reasons for making the distinction while parsing. First, identifying complements is complex enough to warrant a probabilistic treatment. Lexical information is needed — for example, knowledge that "week" is likely to be a temporal modifier. Knowledge about subcategorization preferences — for example that a verb takes exactly one subject — is also required. For example, "week" can sometimes be a subject, as in *Last week was a good one*, so the model must balance the preference for having a subject against

6 Note that this feature means that dynamic programming parsing algorithms for the model must keep track of whether each constituent does or does not have a verb in the string to the right or left of its head. See (Collins 1999) for a full description of the parsing algorithms.
7 In the models described in (Collins 1997), there was a third question concerning punctuation: (3) Does the string contain 0, 1, 2 or > 2 commas? (where a comma is anything tagged as "," or ":"). The model described in this paper has a cleaner incorporation of punctuation into the generative process, as described in section 4.3.
8 Except "IBM" is closer to the `VP`, but note that "IBM" is also the subject in "IBM last week bought Lotus".
9 We use the term *complement* in a broad sense that includes both complements and specifiers under the terminology of Government and Binding.

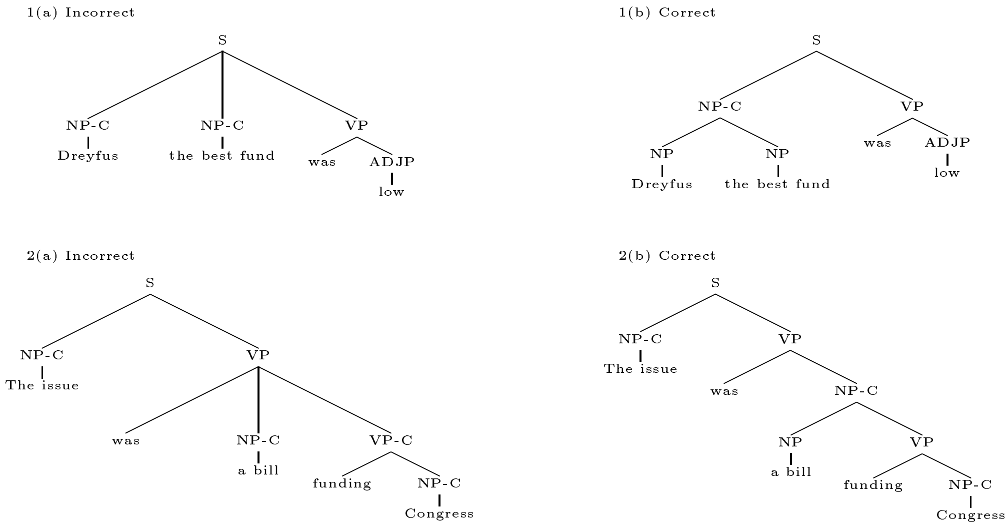


Figure 6
Two examples where the assumption that modifiers are generated independently of each other leads to errors. In (1) the probability of generating both "Dreyfus" and "fund" as subjects, $P(\texttt{NP-C(Dreyfus)} \mid \texttt{S,VP,was}) * P(\texttt{NP-C(fund)} \mid \texttt{S,VP,was})$ is unreasonably high. (2) is similar: $P(\texttt{NP-C(bill),VP-C(funding)} \mid \texttt{VP,VB,was}) =$ $P(\texttt{NP-C(bill)} \mid \texttt{VP,VB,was}) * P(\texttt{VP-C(funding)} \mid \texttt{VP,VB,was})$ is a bad independence assumption.

the relative improbability of "week" being the head-word of a subject. These problems are not restricted to `NP`s, compare "The spokeswoman said (`SBAR` that the asbestos was dangerous)" vs. "Bonds beat short-term investments (`SBAR` because the market is down)", where an `SBAR` headed by "that" is a complement, but an `SBAR` headed by "because" is an adjunct.

A second reason for incorporating the complement/adjunct distinction into the parsing model is that this may help parsing accuracy. The assumption that complements are generated independently of each other often leads to incorrect parses. See figure 6 for examples.

3.2.1 Identifying Complements and Adjuncts in the Penn Treebank We add the `-C` suffix to all non-terminals in training data that satisfy the following conditions:

1. The non-terminal must be: (1) an `NP`, `SBAR`, or `S` whose parent is an `S`; (2) an `NP, SBAR, S`, or `VP` whose parent is a `VP`; or (3) an `S` whose parent is an `SBAR`.

2. The non-terminal must *not* have one of the following semantic tags: `ADV, VOC, BNF, DIR, EXT, LOC, MNR, TMP, CLR` or `PRP`. See (Marcus et al. 94) for an explanation of what these tags signify. For example, the `NP` "Last week" in figure 2 would have the `TMP` (temporal) tag; and the `SBAR` in "(`SBAR` because the market is down)", would have the `ADV` (adverbial) tag.

3. The non-terminal must not be on the RHS of a coordinated phrase. For example, in the rule `S` $\rightarrow$ `S CC S` the two child `S`'s would not be marked as complements.

In addition, the first child following the head of a prepositional phrase is marked as a complement.

3.2.2 Probabilities over Subcategorization Frames Model 1 could be retrained on training data with the enhanced set of non-terminals, and it might learn the lexical properties which distinguish complements and adjuncts ("IBM" vs "week", or "that" vs. "because"). However, it would still suffer from the bad independence assumptions illustrated in figure 6. To solve these kinds of problems, the generative process is extended to include a probabilistic choice of left and right subcategorization frames:

1. Choose a head H with probability $P_h(H \mid P, h)$.
2. Choose left and right subcat frames, LC and RC, with probabilities $P_{lc}(LC \mid P, H, h)$ and $P_{rc}(RC \mid P, H, h)$. Each subcat frame is a multiset[10] specifying the complements that the head requires in its left or right modifiers.
3. Generate the left and right modifiers with probabilities $P_l(L_i(l_i) \mid H, P, h, distance_l(i-1), LC)$ and $P_r(R_i(r_i) \mid H, P, h, distance_r(i-1), RC)$ respectively.

Thus the subcat requirements are added to the conditioning context. As complements are generated they are removed from the appropriate subcat multiset. Most importantly, the probability of generating the `STOP` symbol will be 0 when the subcat frame is *non-empty*, and the probability of generating a complement will be 0 when it is not in the subcat frame; thus all and only the required complements will be generated.

The probability of the phrase `S(bought)` $\rightarrow$ `NP(week) NP-C(IBM) VP(bought)` is now:

$$P_h(\texttt{VP} \mid \texttt{S,bought}) \times P_{lc}(\{\texttt{NP-C}\} \mid \texttt{S,VP,bought}) \times P_{rc}(\{\} \mid \texttt{S,VP,bought}) \times$$
$$P_l(\texttt{NP-C(IBM)} \mid \texttt{S,VP,bought}, \{\texttt{NP-C}\}) \times P_l(\texttt{NP(week)} \mid \texttt{S,VP,bought}, \{\}) \times$$
$$P_l(\texttt{STOP} \mid \texttt{S,VP,bought}, \{\}) \times P_r(\texttt{STOP} \mid \texttt{S,VP,bought}, \{\})$$

Here the head initially decides to take a single `NP-C` (subject) to its left, and no complements to its right. `NP-C(IBM)` is immediately generated as the required subject, and `NP-C` is removed from LC, leaving it empty when the next modifier, `NP(week)` is generated. The incorrect structures in figure 6 should now have low probability because $P_{lc}(\{\texttt{NP-C,NP-C}\} \mid \texttt{S,VP,was})$ and $P_{rc}(\{\texttt{NP-C,VP-C}\} \mid \texttt{VP,VB,was})$ should be small.

3.3 Model 3: Traces and Wh-Movement

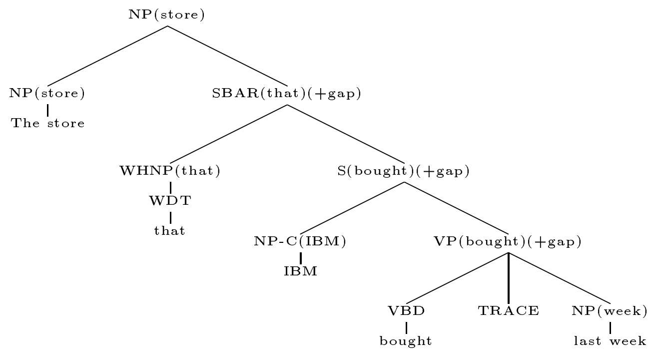
Another obstacle to extracting predicate-argument structure from parse trees is wh-movement. This section describes a probabilistic treatment of extraction from relative clauses. Noun phrases are most often extracted from subject position, object position, or from within `PP`s:

Example
The store (`SBAR` that `TRACE` bought Lotus)

Example
The store (`SBAR` that IBM bought `TRACE`)

Example
The store (`SBAR` that IBM bought Lotus from `TRACE`)

10 A multiset, or bag, is a set which may contain duplicate non-terminal labels.



(1) NP	$\rightarrow$	NP	SBAR(+gap)	
(2) SBAR(+gap)	$\rightarrow$	WHNP	S-C(+gap)	
(3) S(+gap)	$\rightarrow$	NP-C	VP(+gap)	
(4) VP(+gap)	$\rightarrow$	VB	TRACE	NP

Figure 7
A `+gap` feature can be added to non-terminals to describe wh-movement. The top-level `NP` initially generates an `SBAR` modifier, but specifies that it must contain an `NP` trace by adding the *+gap* feature. The gap is then passed down through the tree, until it is discharged as a `TRACE` complement to the right of *bought*.

It might be possible to write rule-based patterns that identify traces in a parse tree. However, we argue again that this task is best integrated into the parser: the task is complex enough to warrant a probabilistic treatment, and integration may help parsing accuracy. A couple of complexities are that modification by an `SBAR` does not always involve extraction (e.g., "the fact (`SBAR` that besoboru is played with a ball and a bat)"), and it is not uncommon for extraction to occur through several constituents, (e.g., "The changes (`SBAR` that he said the government was prepared to make `TRACE`)").

The second reason for an integrated treatment of traces is to improve the parameterization of the model. In particular, the subcategorization probabilities are smeared by extraction. In examples 1, 2 and 3 above 'bought' is a transitive verb; but without knowledge of traces, example 2 in training data will contribute to the probability of 'bought' being an intransitive verb.

Formalisms similar to GPSG (Gazdar et al. 1995) handle wh-movement by adding a *gap* feature to each non-terminal in the tree, and propagating gaps through the tree until they are finally discharged as a trace complement (see figure 7). In extraction cases the Penn treebank annotation co-indexes a `TRACE` with the `WHNP` head of the `SBAR`, so it is straightforward to add this information to trees in training data.

Given that the LHS of the rule has a gap, there are 3 ways that the gap can be passed down to the RHS:

Head The gap is passed to the head of the phrase, as in rule (3) in figure 7.

Left, Right The gap is passed on recursively to one of the left or right modifiers of the head, or is discharged as a `TRACE` argument to the left/right of the head. In rule (2) it is passed on to a right modifier, the `S` complement. In rule (4) a `TRACE` is generated to the right of the head `VB`.

We specify a parameter type $P_g(G \mid P, h, H)$ where G is either **Head**, **Left** or **Right**. The generative process is extended to choose between these cases after generating the

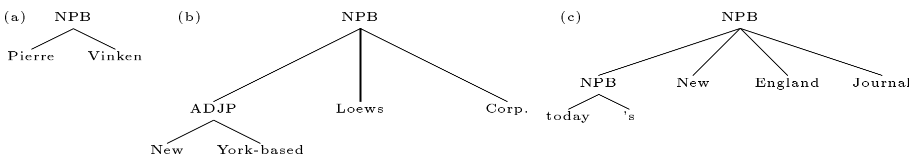


Figure 8
Three examples of structures with base-NPs

head of the phrase. The rest of the phrase is then generated in different ways depending on how the gap is propagated. In the **Head** case the left and right modifiers are generated as normal. In the **Left, Right** cases a `+gap` requirement is added to either the left or right SUBCAT variable. This requirement is fulfilled (and removed from the subcat list) when either a trace, or a modifier non-terminal which has the `+gap` feature, is generated. For example, Rule (2), `SBAR(that)(+gap)` $\rightarrow$ `WHNP(that) S-C(bought)(+gap)`, has probability

$$P_h(\texttt{WHNP} \mid \texttt{SBAR,that}) \times P_g(\texttt{Right} \mid \texttt{SBAR,WHNP,that}) \times P_{lc}(\{\} \mid \texttt{SBAR,WHNP,that}) \times$$
$$P_{rc}(\{\texttt{S-C}\} \mid \texttt{SBAR,WHNP,that}) \times P_r(\texttt{S-C(bought)(+gap)} \mid \texttt{SBAR,WHNP,that}, \{\texttt{S-C,+gap}\}) \times$$
$$P_r(\texttt{STOP} \mid \texttt{SBAR,WHNP,that}, \{\}) \times P_l(\texttt{STOP} \mid \texttt{SBAR,WHNP,that}, \{\})$$

Rule (4), `VP(bought)(+gap)` $\rightarrow$ `VB(bought) TRACE NP(week)`, has probability

$$P_h(\texttt{VB} \mid \texttt{VP,bought}) \times P_g(\texttt{Right} \mid \texttt{VP,bought,VB}) \times P_{lc}(\{\} \mid \texttt{VP,bought,VB}) \times$$
$$P_{rc}(\{\texttt{NP-C}\} \mid \texttt{VP,bought,VB}) \times P_r(\texttt{TRACE} \mid \texttt{VP,bought,VB}, \{\texttt{NP-C, +gap}\}) \times$$
$$P_r(\texttt{NP(week)} \mid \texttt{VP,bought,VB}, \{\}) \times P_l(\texttt{STOP} \mid \texttt{VP,bought,VB}, \{\}) \times$$
$$P_r(\texttt{STOP} \mid \texttt{VP,bought,VB}, \{\})$$

In rule (2) `Right` is chosen, so the `+gap` requirement is added to RC. Generation of `S-C(bought)(+gap)` fulfills both the `S-C` and `+gap` requirements in RC. In rule (4) `Right` is chosen again. Note that generation of `TRACE` satisfies both the `NP-C` and `+gap` subcat requirements.

4 Special Cases: Linguistically Motivated Refinements to the Models

Sections 3.1 to 3.3 described the basic framework for the parsing models in this paper. In this section we describe how some linguistic phenomena (non-recursive `NP`s, and coordination, for example) clearly violate the independence assumptions assumed in the general models. We describe a number of these special cases, in each case arguing how the phenomenon violates the independence assumptions, and then describing how the model can be refined to deal with the problem.

4.1 Non-recursive NPs

We define non-recursive `NP`s (from here on referred to as base-NPs, and labeled `NPB` rather than `NP`) as `NP`s that do not directly dominate an `NP` themselves, unless the dominated `NP` is a possessive `NP` (i.e. it directly dominates a POS-tag `POS`). Figure 8 gives some examples. Base-NPs deserve special treatment for three reasons:

- The boundaries of base-NPs are often strongly marked. In particular, the start points of base-NPs are often marked with a determiner or another distinctive item such as an adjective. Because of this, the probability of generating the `STOP` symbol should be greatly increased when the previous modifier is, for example, a determiner. As they stand, the independence assumptions in the three models lose this information. The probability of `NPB(dog)` $\rightarrow$ `DT(the)`

NN(dog) would be estimated as[11]

$$P_h(\texttt{NN} \mid \texttt{NPB,dog}) \times \mathcal{P}_l(\texttt{DT(the)} \mid \texttt{NPB,NN,dog}) \times$$
$$P_l(\texttt{STOP} \mid \texttt{NPB,NN,dog}) \times P_r(\texttt{STOP} \mid \texttt{NPB,NN,dog})$$

In making the independence assumption

$$P_l(\texttt{STOP} \mid \texttt{DT(the)},\texttt{NPB,NN,dog}) = P_l(\texttt{STOP} \mid \texttt{NPB,NN,dog})$$

the model will fail to learn that the STOP symbol is very likely to follow a determiner. As a result, the model will assign unreasonably high probability to NPs such as [NP *yesterday the dog*] in sentences such as [*yesterday the dog barked*].

- The annotation standard in the treebank leaves the internal structure of base-NPs underspecified. For example, both *pet food volume* (where *pet* modifies *food* and *food* modifies *volume*) and *vanilla ice cream* (where both *vanilla* and *ice* modify *cream*) would have the structure NPB $\rightarrow$ NN NN NN. Because of this, there is no reason to believe that modifiers within NPBs are dependent on the head rather than the previous modifier. In fact, if it so happened that a majority of phrases were like *pet food volume*, then conditioning on the previous modifier rather than the head would be preferable.

- In general it is important (in particular for the distance measure to be effective) to have different non-terminal labels for what are effectively different X-bar levels. See section 7.3.2 for further discussion.

For these reasons the following modifications were made to the models:

- The non-terminal label for base-NPs is changed from NP to NPB. For consistency, whenever an NP is seen with no pre or post modifiers, an NPB level is added. For example, [S [NP the dog] [VP barks]] would be transformed to [S [NP [NPB the dog]] [VP barks]]. These "extra" NPBs are removed before scoring the output of the parser against the treebank.

- The independence assumptions are different when the parent non-terminal is an NPB. Specifically, equations 5 and 6 are modified to be

$$\begin{aligned} P_l(L_i(l_i) \mid H, P, h, L_1(l_1)...L_{i-1}(l_{i-1})) &= \mathcal{P}_l(L_i(l_i) \mid P, L_{i-1}(l_{i-1})) \\ P_r(R_i(r_i) \mid H, P, h, R_1(r_1)...R_{i-1}(r_{i-1})) &= P_r(R_i(r_i) \mid P, R_{i-1}(r_{i-1})) \end{aligned}$$

  The modifier and previous-modifier non-terminals are always adjacent, so the distance variable is constant and is omitted. For the purposes of this model, $L_0(l_0)$ and $R_0(r_0)$ are defined to be $H(h)$. The probability of the previous example is now

$$P_h(\texttt{NN} \mid \texttt{NPB,dog}) \times \mathcal{P}_l(\texttt{DT(the)} \mid \texttt{NPB,NN,dog}) \times$$
$$P_l(\texttt{STOP} \mid \texttt{NPB,DT,the}) \times P_r(\texttt{STOP} \mid \texttt{NPB,NN,dog})$$

  Presumably $P_l(\texttt{STOP} \mid \texttt{NPB,DT,the})$ will be very close to 1.

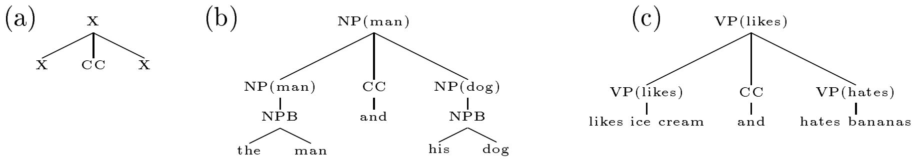


Figure 9
(a) The generic way of annotating coordination in the treebank. (b) and (c) show specific examples (with base-NPs added as described in section 4.1). Note that the first item of the conjunct is taken as the head of the phrase.

4.2 Coordination

Coordination constructions are another example where the independence assumptions in the basic models fail badly (at least given the current annotation method in the treebank). Figure 9 shows how coordination is annotated in the treebank.[12] To use an example to illustrate the problems, take the rule `NP(man)` $\rightarrow$ `NP(man) CC(and) NP(dog)`, which has probability

$$P_h(\texttt{NP} \mid \texttt{NP,man}) \times P_l(\texttt{STOP} \mid \texttt{NP,NP,man}) \times P_r(\texttt{CC(and)} \mid \texttt{NP,NP,man}) \times$$
$$P_r(\texttt{NP(dog)} \mid \texttt{NP,NP,man}) \times P_r(\texttt{STOP} \mid \texttt{NP,NP,man})$$

The independence assumptions mean that the model fails to learn that there is always exactly one phrase following the coordinator (`CC`). The basic probability models will give much too high probability to unlikely phrases such as `NP` $\rightarrow$ `NP CC` or `NP` $\rightarrow$ `NP CC NP NP`. For this reason we alter the generative process to allow generation of both the coordinator and the following phrase in one step; instead of just generating a non-terminal at each step, a non-terminal and a binary-valued `coord` flag are generated. `coord`=1 if there is a coordination relationship. In the generative process, generation of a `coord=1` flag along with a modifier triggers an additional step in the generative process, namely the generation of the coordinator tag/word pair, parameterized by the P_{cc} parameter. For the preceding example this would give probability

$$P_h(\texttt{NP} \mid \texttt{NP,man}) \times \mathcal{P}_l(\texttt{STOP} \mid \texttt{NP,NP,man}) \times P_r(\texttt{NP(dog), coord=1} \mid \texttt{NP,NP,man}) \times$$
$$P_r(\texttt{STOP} \mid \texttt{NP,NP,man}) \times P_{cc}(\texttt{CC,and} \mid \texttt{NP,NP,NP,man,dog})$$

Note the new type of parameter, P_{cc}, for the generation of the coordinator word and POS-tag. The generation of `coord=1` along with `NP(dog)` in the example implicitly requires generation of a coordinator tag/word pair through the P_{cc} parameter. The generation of this tag/word pair is conditioned on the two words in the coordination dependency (`man` and `dog` in the example), and the label on their relationship (`NP,NP,NP` in the example, representing `NP` coordination).

The `coord` flag is implicitly 0 when normal non-terminals are generated, for example the phrase `S(bought)` $\rightarrow$ `NP(week) NP(IBM) VP(bought)` now has probability

$$P_h(\texttt{VP} \mid \texttt{S,bought}) \times P_l(\texttt{NP(IBM),coord=0} \mid \texttt{S,VP,bought}) \times$$
$$P_l(\texttt{NP(week),coord=0} \mid \texttt{S,VP,bought}) \times P_l(\texttt{STOP} \mid \texttt{S,VP,bought}) \times$$
$$P_r(\texttt{STOP} \mid \texttt{S,VP,bought})$$

11 For simplicity, we give probability terms under Model 1 with no distance variables – the probability terms with distance variables, or for Models 2 and 3, will be similar, but with the addition of various pieces of conditioning information.
12 See Appendix A of (Collins 1999) for a description of how the head rules treat phrases involving coordination.

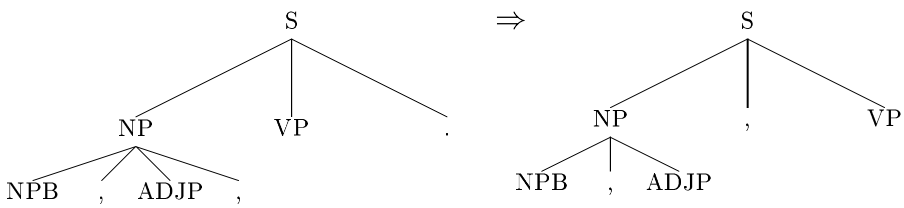


Figure 10
A parse tree before and after the punctuation transformations.

4.3 Punctuation

This section describes our treatment of "punctuation" in the model, where "punctuation" is used to refer to words tagged as a comma or colon. Previous work — the models described in (Collins 1996) and the earlier version of these generative models described in (Collins 1997) — conditioned on punctuation as surface features of the string, treating it quite differently from lexical items. In particular, the model in (Collins 1997) failed to generate punctuation, a deficiency of the model. This section describes how punctuation is integrated into the generative models.

Our first step is to raise punctuation as high in the parse trees as possible. Punctuation at the beginning or end of sentences is removed from the training/test data altogether.[13] All punctuation items apart from those tagged as comma or colon (items such as quotes and periods, tagged **" "** or **.**) are removed altogether. These transformations mean that punctuation always appears between two non-terminals, as opposed to appearing at the end of a phrase. See figure 10 for an example.

Punctuation is then treated in a very similar way to coordination: our intuition is that there is a strong dependency between the punctuation mark and the modifier generated after it. Punctuation is therefore generated with the following phrase through a `punc` flag which is similar to the `coord` flag (a binary-valued feature equal to 1 if a punctuation mark is generated with the following phrase).

Under this model, `NP(Vinken)` $\rightarrow$ `NPB(Vinken) ,(,) ADJP(old)` would have probability

$$\begin{aligned} & P_h(\texttt{NPB} \mid \texttt{NP,Vinken}) \times P_l(\texttt{STOP} \mid \texttt{NP,NPB,Vinken}) \times \\ & P_r(\texttt{ADJP(old),coord=0,punc=1} \mid \texttt{NP,NPB,Vinken}) \times \\ & P_r(\texttt{STOP} \mid \texttt{NP,NPB,bought}) \times P_p(\texttt{,}\ \ \texttt{,} \mid \texttt{NP,NPB,ADJP,Vinken,old}) \end{aligned} \quad (7)$$

P_p is a new parameter type for generation of punctuation tag/word pairs. The generation of `punc=1` along with `ADJP(old)` in the example implicitly requires generation of a punctuation tag/word pair through the P_p parameter. The generation of this tag/word pair is conditioned on the two words in the punctuation dependency (`Vinken` and `old` in the example), and the label on their relationship (`NP,NPB,ADJP` in the example.)

4.4 Sentences with empty (PRO) subjects

Sentences in the treebank occur frequently with PRO subjects which may or may not be controlled: as the treebank annotation currently stands the non-terminal is `S` whether or not a sentence has an overt subject. This is a problem for the subcategorization probabilities in models 2 and 3 — the probability of having zero subjects, $\mathcal{P}_{lc}(\{\} \mid \texttt{S, VP,}\ verb)$,

13 As one of the anonymous reviewers of this paper pointed out, this choice of discarding the sentence-final punctuation may not be optimal, as the final punctuation mark may well carry useful information about the sentence structure.

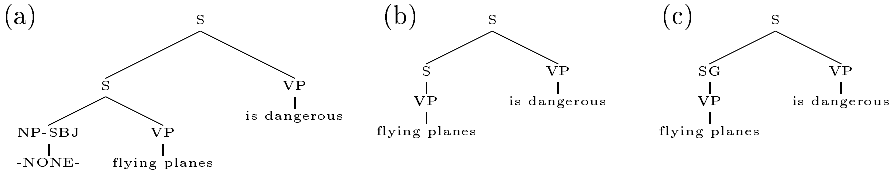


Figure 11
(a) the treebank annotates sentences with empty subjects with an empty *-NONE-* element under subject position; (b) in training (and for evaluation), this null element is removed; (c) in models 2 and 3 sentences without subjects are changed to have a non-terminal `SG`.

Back-off Level	$P_h(H \mid ...)$	$P_g(G \mid ...)$ $P_{lc}(LC \mid ...)$ $P_{rc}(RC \mid ...)$	$P_{L1}(L_i(lt_i), c, p \mid ...)$ $P_{R1}(R_i(rt_i), c, p \mid ...)$	$P_{L2}(lw_i \mid ...)$ $P_{R2}(rw_i \mid ...)$
1	P, w, t	P, H, w, t	P, H, w, t, Δ, LC	L_i, lt_i, c, p, P, H, w, t, Δ, LC
2	P, t	P, H, t	P, H, t, Δ, LC	L_i, lt_i, c, p, P, H, t, Δ, LC
3	P	P, H	P, H, Δ, LC	lt_i

Table 1
The conditioning variables for each level of back-off. For example, P_h estimation interpolates $e_1 = P_h(H \mid P, w, t)$, $e_2 = P_h(H \mid P, t)$, and $e_3 = P_h(H \mid P)$. Δ is the distance measure.

will be fairly high because of this. In addition, sentences with and without subjects appear in quite different syntactic environments. For these reasons we modify the non-terminal for sentences without subjects to be `SG`. See figure 11. The resulting model has a cleaner division of subcategorization: $P_{lc}(\{\texttt{NP-C}\} \mid \texttt{S, VP, } verb) \approx 1$ and $P_{lc}(\{\texttt{NP-C}\} \mid \texttt{SG, VP, } verb) = 0$. The model will learn probabilistically the environments in which `S` and `SG` are likely to appear.

4.5 A Punctuation Constraint

As a final step, we use the rule concerning punctuation introduced in (Collins 1996). The constraint is as follows. If for any constituent `Z` in the chart `Z` $\rightarrow$ `<..X Y..>` two of its children `X` and `Y` are separated by a comma, then the last word in `Y` must be directly followed by a comma, or must be the last word in the sentence. In training data 96% of commas follow this rule. The rule has the benefit of improving efficiency by reducing the number of constituents in the chart. It would be preferable to develop a probabilistic analogue of this rule, but we leave this to future research.

5 Practical Issues

5.1 Parameter Estimation

Table 1 shows the various levels of back-off for each type of parameter in the model. Note that we decompose $\mathcal{P}_L(L_i(lw_i, lt_i), c, p \mid P, H, w, t, \Delta, LC)$ (where lw_i and lt_i are the word and POS tag generated with non-terminal L_i, c and p are the `coord` and `punc` flags associated with the non-terminal, Δ is the distance measure) into the product

$$P_{L1}(L_i(lt_i), c, p \mid P, H, w, t, \Delta, LC) \times P_{L2}(lw_i \mid L_i, lt_i, c, p, P, H, w, t, \Delta, LC)$$

These two probabilities are then smoothed separately. (Eisner 1996b) originally used POS tags to smooth a generative model in this way. In each case the final estimate is

$$e = \lambda_1 e_1 + (1 - \lambda_1)(\lambda_2 e_2 + (1 - \lambda_2) e_3)$$

where e_1, e_2 and e_3 are maximum likelihood estimates with the context at levels 1, 2 and 3 in the table, and λ_1, λ_2 and λ_3 are smoothing parameters where $0 \leq \lambda_i \leq 1$. We use the smoothing method described in (Bikel et al. 1997), which is derived from a method described in (Witten and Bell 1991). First, say the most specific estimate $e_1 = \frac{n_1}{f_1}$ – that is, f_1 is the value of the denominator count in the relative frequency estimate. Second, define u_1 to be the number of distinct outcomes seen in the f_1 events in training data. The variable u_1 can take any value from 1 to f_1 inclusive. Then we set

$$\lambda_1 = \frac{f_1}{f_1 + 5u_1}$$

Analogous definitions for f_2 and u_2 lead to $\lambda_2 = \frac{f_2}{f_2+5u_2}$. The constant 5 was chosen to maximize accuracy on the development set, section 0 of the treebank (in practice it was found that any value in the range 2–5 gave a very similar level of performance).

5.2 Unknown Words and Part of Speech Tagging

All words occurring less than 6 times[14] in training data, and words in test data which have never been seen in training, are replaced with the `UNKNOWN` token. This allows the model to robustly handle the statistics for rare or new words. Words in test data that have not been seen in training are deterministically assigned the POS tag which is assigned by the tagger described in (Ratnaparkhi 1996). As a preprocessing step the tagger is used to decode each test data sentence. All other words are tagged during parsing, the output from Ratnaparkhi's tagger being ignored. The POS tags allowed for each word are limited to those which have been seen in training data for that word (any word/tag pairs not seen in training would give an estimate of zero in the P_{L2} and P_{R2} distributions). The model is fully integrated, in that part of speech tags are statistically generated along with words in the models, so that the parser will make a statistical decision as to the most likely tag for each known word in the sentence.

5.3 The Parsing Algorithm

The parsing algorithm for the models is a dynamic programming algorithm, which is very similar to standard chart parsing algorithms for probabilistic or weighted grammars. The algorithm has complexity $O(n^5)$, where n is the number of words in the string. In practice, pruning strategies (methods which discard lower probability constituents in the chart) can improve efficiency a great deal. The appendices of (Collins 1999) give a precise description of the parsing algorithms, an analysis of their computational complexity, and also a description of the pruning methods that are employed.

See (Eisner and Satta 1999) for an $O(n^4)$ algorithm for lexicalized grammars, which could be applied to the models in this paper. (Eisner and Satta 1999) also describe an $O(n^3)$ algorithm for a restricted class of lexicalized grammars, it is an open question whether this restricted class includes the models in this paper.

6 Results

The parser was trained on sections 02–21 of the Wall Street Journal portion of the Penn Treebank (Marcus et al. 1993) (approximately 40,000 sentences), and tested on section 23 (2,416 sentences). We use the PARSEVAL measures (Black et al. 1991) to compare performance:

$$\textbf{Labeled Precision} = \frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in proposed parse}}$$

14 In (Collins 1999) we erroneously stated that all words occuring less than 5 times in training data were classified as being “unknown”: thanks to Dan Bikel for pointing out this error.

MODEL	≤ 40 Words (2245 sentences)				
	LR	LP	CBs	0 CBs	≤ 2 CBs
(Magerman 1995)	84.6%	84.9%	1.26	56.6%	81.4%
(Collins 1996)	85.8%	86.3%	1.14	59.9%	83.6%
(Goodman 1997)	84.8%	85.3%	1.21	57.6%	81.4%
(Charniak 1997)	87.5%	87.4%	1.00	62.1%	86.1%
Model 1	87.9%	88.2%	0.95	65.8%	86.3%
Model 2	88.5%	88.7%	0.92	66.7%	87.1%
Model 3	88.6%	88.7%	0.90	67.1%	87.4%
(Charniak 2000)	90.1%	90.1%	0.74	70.1%	89.6%
(Collins 2000)	90.1%	90.4%	0.73	70.7%	89.6%

MODEL	≤ 100 Words (2416 sentences)				
	LR	LP	CBs	0 CBs	≤ 2 CBs
(Magerman 1995)	84.0%	84.3%	1.46	54.0%	78.8%
(Collins 1996)	85.3%	85.7%	1.32	57.2%	80.8%
(Charniak 1997)	86.7%	86.6%	1.20	59.5%	83.2%
(Ratnaparkhi 1997)	86.3%	87.5%	1.21	60.2%	—
Model 1	87.5%	87.7%	1.09	63.4%	84.1%
Model 2	88.1%	88.3%	1.06	64.0%	85.1%
Model 3	88.0%	88.3%	1.05	64.3%	85.4%
(Charniak 2000)	89.6%	89.5%	0.88	67.6%	87.7%
(Collins 2000)	89.6%	89.9%	0.87	68.3%	87.7%

Table 2
Results on Section 23 of the WSJ Treebank. **LR/LP** = labeled recall/precision. **CBs** is the average number of crossing brackets per sentence. **0 CBs,** ≤ 2 **CBs** are the percentage of sentences with 0 or ≤ 2 crossing brackets respectively. All the results in this table are for models trained and tested on the same data, using the same evaluation metric. (Note that these results show a slight improvement over those in (Collins 97); the main model changes were the improved treatment of punctuation (section 4.3) together with the addition of the P_p and P_{cc} parameters.)

$$\textbf{Labeled Recall} = \frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in treebank parse}}$$

Crossing Brackets = number of constituents which violate constituent boundaries with a constituent in the treebank parse.

For a constituent to be 'correct' it must span the same set of words (ignoring punctuation, i.e. all tokens tagged as commas, colons or quotes) and have the same label[15] as a constituent in the treebank parse. Table 2 shows the results for Models 1, 2 and 3, and a variety of other models in the literature. Two models (Collins 2000; Charniak 2000) outperform models 2 and 3 on section 23 of the treebank. (Collins 2000) uses a technique based on boosting algorithms for machine learning, which reranks n-best output from model 2 in this paper. (Charniak 2000) describes a series of enhancements to the earlier model of (Charniak 1997).

The precision/recall of the traces found by Model 3 was 93.8%/90.1% (out of 437 cases in section 23 of the treebank), where three criteria must be met for a trace to be

15 (Magerman 1995) collapses ADVP and PRT to the same label, for comparison we also removed this distinction when calculating scores.

"correct": (1) it must be an argument to the correct head-word; (2) it must be in the correct position in relation to that head word (preceding or following); (3) it must be dominated by the correct non-terminal label. For example, in figure 7 the trace is an argument to **bought**, which it **follows**, and it is dominated by a **VP**. Of the 437 cases, 341 were string-vacuous extraction from subject position, recovered with 96.3%/98.8% precision/recall; and 96 were longer distance cases, recovered with 81.4%/59.4% precision/recall [16].

7 Discussion

This section discusses some aspects of the models in more detail. Section 7.1 gives a much more detailed analysis of the parsers' performance. In section 7.2 we discuss the distance features in the model. In section 7.3 we discuss how the model interacts with the Penn Treebank style of annotation. Finally, in section 7.4 we discuss the need to break down context-free rules in the treebank in such a way that the model will generalize to give non-zero probability to rules not seen in training. In each case we use three methods of analysis. First, we can consider how various aspects of the model affect parsing performance, through accuracy measurements on the treebank. Second, we can look at the frequency of different constructions in the treebank. Third, we can consider linguistically motivated examples as a way of justifying various modeling choices.

7.1 A Closer look at the Results

In this section we look more closely at the parser, by evaluating its performance on specific constituents or constructions. The intention is to get a better idea of the parser's strengths and weaknesses. First, table 3 has a breakdown of precision and recall by constituent type. Although somewhat useful in understanding parser performance, a breakdown of accuracy by constituent type fails to capture the idea of attachment accuracy. For this reason we also evaluate the parser's precision and recall in recovering dependencies between words. This gives a better indication of the accuracy on different kinds of attachments. A dependency is defined as a triple with the following elements (see figure 12 for an example tree and its associated dependencies):

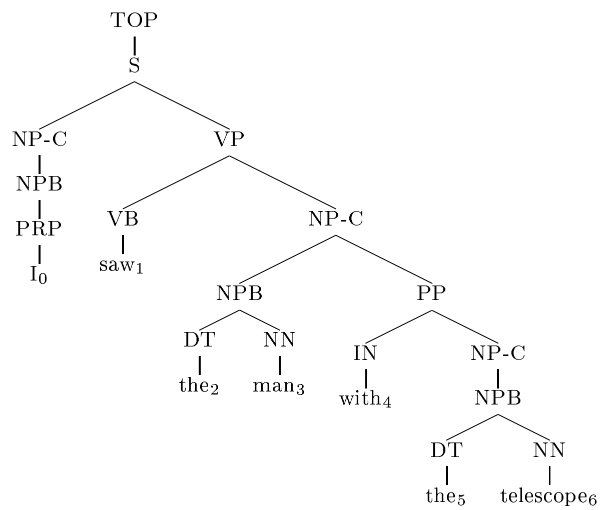
1) Modifier The index of the modifier word in the sentence.

2) Head The index of the head word in the sentence.

3) Relation A ⟨`Parent`, `Head`, `Modifier`, `Direction`⟩ 4-tuple, where the four elements are the parent, head and modifier non-terminals involved in the dependency, and the direction of the dependency (L for left, R for right). For example, ⟨`S`, `VP`, `NP-C`, `L`⟩ would indicate a subject-verb dependency. In coordination cases there is a fifth element of the tuple, `CC`. For example, ⟨`NP`, `NP`, `NP`, `R`, `CC`⟩ would be an instance of NP coordination.

In addition, the relation is "normalized" to some extent. First, all POS tags are replaced with the token `TAG`: this is so that POS tagging errors do not lead to errors in

16 We exclude infinitival relative clauses from these figures, for example "I called a plumber `TRACE` to fix the sink" where 'plumber' is co-indexed with the trace subject of the infinitival. The algorithm scored 41%/18% precision/recall on the 60 cases in section 23 — but infinitival relatives are extremely difficult even for human annotators to distinguish from purpose clauses (in this case, the infinitival could be a purpose clause modifying 'called') (Ann Taylor, p.c.)



"Raw" Dependencies			Normalized Dependencies		
Relation	Modifier	Head	Relation	Modifier	Head
`S VP NP-C L`	0	1	`S VP NP-C L`	0	1
`TOP TOP S R`	1	-1	`TOP TOP S R`	1	-1
`NPB NN DT L`	2	3	`NPB TAG TAG L`	2	3
`VP VB NP-C R`	3	1	`VP TAG NP-C R`	3	1
`NP-C NPB PP R`	4	3	`NP NPB PP R`	4	3
`NPB NN DT L`	5	6	`NPB TAG TAG L`	5	6
`PP IN NP-C R`	6	4	`PP TAG NP-C R`	6	4

Figure 12
A tree and its associated dependencies. Note that in "normalizing" dependencies, all POS tags are replaced with `TAG`, and the NP-C parent in the fifth relation is replaced with NP.

Evaluation	Precision	Recall
No Labels	91.0%	90.9%
No Complements	88.5%	88.5%
All	88.3%	88.3%

Figure 13
Dependency accuracy on Section 0 of the treebank with Model 2. *No Labels* means that only the dependency needs to be correct, the relation may be wrong; *No Complements* means all complement (-C) markings are stripped before comparing relations; *All* means complement markings are kept on the modifying non-terminal.

Proportion	Count	Label	Recall	Precision
42.21	15146	NP	91.15	90.26
19.78	7096	VP	91.02	91.11
13.00	4665	S	91.21	90.96
12.83	4603	PP	86.18	85.51
3.95	1419	SBAR	87.81	88.87
2.59	928	ADVP	82.97	86.52
1.63	584	ADJP	65.41	68.95
1.00	360	WHNP	95.00	98.84
0.92	331	QP	84.29	78.37
0.48	172	PRN	32.56	61.54
0.35	126	PRT	86.51	85.16
0.31	110	SINV	83.64	88.46
0.27	98	NX	12.24	66.67
0.25	88	WHADVP	95.45	97.67
0.08	29	NAC	48.28	63.64
0.08	28	FRAG	21.43	46.15
0.05	19	WHPP	100.00	100.00
0.04	16	UCP	25.00	28.57
0.04	16	CONJP	56.25	69.23
0.04	15	SQ	53.33	66.67
0.03	12	SBARQ	66.67	88.89
0.03	9	RRC	11.11	33.33
0.02	7	LST	57.14	100.00
0.01	3	X	0.00	—
0.01	2	INTJ	0.00	—

Table 3
Recall and precision for different constituent types, for section 0 of the treebank with model 2. Label is the non-terminal label; Proportion is the percentage of constituents in the treebank section 0 that have this label; Count is the number of constituents that have this label.

dependencies[17]. Second, any complement markings on the parent or head non-terminal are removed. For example, $\langle \texttt{NP-C}, \texttt{NPB}, \texttt{PP}, \texttt{R} \rangle$ is replaced by $\langle \texttt{NP}, \texttt{NPB}, \texttt{PP}, \texttt{R} \rangle$. This prevents parsing errors where a complement has been mistaken to be an adjunct (or vice versa) leading to more than one dependency error. As an example, in figure 12, if the `NP` *the man with the telescope* was mistakenly identified as an adjunct then without normalisation this would lead to two dependency errors: both the `PP` dependency and the verb-object relation would be incorrect. With normalisation, only the verb-object relation is incorrect.

Under this definition, gold-standard and parser-output trees can be converted to sets of dependencies, and precision/recall can be calculated on these dependencies. Dependency accuracies are given for section 0 of the treebank in figure 13. Table 4 gives a breakdown of the accuracies by dependency type.

Tables 5 and 6 show the dependency accuracy for 8 sub-types of dependency, which together account for 94% of all dependencies. These sub-types are:

Complement to a verb: 93.76/92.96 recall/precision. This type includes any re-

17 The justification for this is that there is an estimated 3% error rate in the hand-assigned POS tags in the treebank (Ratnaparkhi 1996), and we didn't want this noise to contribute to dependency errors.

R	CP	P	Count	Relation	Rec	Prec
1	29.65	29.65	11786	NPB TAG TAG L	94.60	93.46
2	40.55	10.90	4335	PP TAG NP-C R	94.72	94.04
3	48.72	8.17	3248	S VP NP-C L	95.75	95.11
4	54.03	5.31	2112	NP NPB PP R	84.99	84.35
5	59.30	5.27	2095	VP TAG NP-C R	92.41	92.15
6	64.18	4.88	1941	VP TAG VP-C R	97.42	97.98
7	68.71	4.53	1801	VP TAG PP R	83.62	81.14
8	73.13	4.42	1757	TOP TOP S R	96.36	96.85
9	74.53	1.40	558	VP TAG SBAR-C R	94.27	93.93
10	75.83	1.30	518	QP TAG TAG R	86.49	86.65
11	77.08	1.25	495	NP NPB NP R	74.34	75.72
12	78.28	1.20	477	SBAR TAG S-C R	94.55	92.04
13	79.48	1.20	476	NP NPB SBAR R	79.20	79.54
14	80.40	0.92	367	VP TAG ADVP R	74.93	78.57
15	81.30	0.90	358	NPB TAG NPB L	97.49	92.82
16	82.18	0.88	349	VP TAG TAG R	90.54	93.49
17	82.97	0.79	316	VP TAG SG-C R	92.41	88.22
18	83.70	0.73	289	NP NP NP R CC	55.71	53.31
19	84.42	0.72	287	S VP PP L	90.24	81.96
20	85.14	0.72	286	SBAR WHNP SG-C R	90.56	90.56
21	85.79	0.65	259	VP TAG ADJP R	83.78	80.37
22	86.43	0.64	255	S VP ADVP L	90.98	84.67
23	86.95	0.52	205	NP NPB VP R	77.56	72.60
24	87.45	0.50	198	ADJP TAG TAG L	75.76	70.09
25	87.93	0.48	189	NPB TAG TAG R	74.07	75.68
26	88.40	0.47	187	VP TAG NP R	66.31	74.70
27	88.85	0.45	180	VP TAG SBAR R	74.44	72.43
28	89.29	0.44	174	VP VP VP R CC	74.14	72.47
29	89.71	0.42	167	NPB TAG ADJP L	65.27	71.24
30	90.11	0.40	159	VP TAG SG R	60.38	68.57
31	90.49	0.38	150	VP TAG S-C R	74.67	78.32
32	90.81	0.32	129	S S S R CC	72.09	69.92
33	91.12	0.31	125	PP TAG SG-C R	94.40	89.39
34	91.43	0.31	124	QP TAG TAG L	77.42	83.48
35	91.72	0.29	115	S VP TAG L	86.96	90.91
36	92.00	0.28	110	NPB TAG QP L	80.91	81.65
37	92.27	0.27	106	SINV VP NP R	88.68	95.92
38	92.53	0.26	104	S VP S-C L	93.27	78.86
39	92.79	0.26	102	NP NP NP R	30.39	25.41
40	93.02	0.23	90	ADJP TAG PP R	75.56	78.16
41	93.24	0.22	89	TOP TOP SINV R	96.63	94.51
42	93.45	0.21	85	ADVP TAG TAG L	74.12	73.26
43	93.66	0.21	83	SBAR WHADVP S-C R	97.59	98.78
44	93.86	0.20	81	S VP SBAR L	88.89	85.71
45	94.06	0.20	79	VP TAG ADVP L	51.90	49.40
46	94.24	0.18	73	SINV VP S L	95.89	92.11
47	94.40	0.16	63	NP NPB SG R	88.89	81.16
48	94.55	0.15	58	S VP PRN L	25.86	48.39
49	94.70	0.15	58	NX TAG TAG R	10.34	75.00
50	94.83	0.13	53	NP NPB PRN R	45.28	60.00

Table 4
Accuracy of the 50 most frequent dependency types in section 0 of the treebank, as recovered by model 2. R = rank; CP = cumulative percentage; P = percentage; Rec = Recall; Prec = precision

Type	Sub-type	Description	Count	Recall	Precision
Complement to a verb	S VP NP-C L	Subject	3248	95.75	95.11
	VP TAG NP-C R	Object	2095	92.41	92.15
6495 = 16.3% of all cases	VP TAG SBAR-C R		558	94.27	93.93
	VP TAG SG-C R		316	92.41	88.22
	VP TAG S-C R		150	74.67	78.32
	S VP S-C L		104	93.27	78.86
	S VP SG-C L		14	78.57	68.75
	...				
	TOTAL		6495	93.76	92.96
Other complements	PP TAG NP-C R		4335	94.72	94.04
	VP TAG VP-C R		1941	97.42	97.98
7473 = 18.8% of all cases	SBAR TAG S-C R		477	94.55	92.04
	SBAR WHNP SG-C R		286	90.56	90.56
	PP TAG SG-C R		125	94.40	89.39
	SBAR WHADVP S-C R		83	97.59	98.78
	PP TAG PP-C R		51	84.31	70.49
	SBAR WHNP S-C R		42	66.67	84.85
	SBAR TAG SG-C R		23	69.57	69.57
	PP TAG S-C R		18	38.89	63.64
	SBAR WHPP S-C R		16	100.00	100.00
	S ADJP NP-C L		15	46.67	46.67
	PP TAG SBAR-C R		15	100.00	88.24
	...				
	TOTAL		7473	94.47	94.12
PP modification	NP NPB PP R		2112	84.99	84.35
	VP TAG PP R		1801	83.62	81.14
4473 = 11.2% of all cases	S VP PP L		287	90.24	81.96
	ADJP TAG PP R		90	75.56	78.16
	ADVP TAG PP R		35	68.57	52.17
	NP NP PP R		23	0.00	0.00
	PP PP PP L		19	21.05	26.67
	NAC TAG PP R		12	50.00	100.00
	...				
	TOTAL		4473	82.29	81.51
Coordination	NP NP NP R		289	55.71	53.31
	VP VP VP R		174	74.14	72.47
763 = 1.9% of all cases	S S S R		129	72.09	69.92
	ADJP TAG TAG R		28	71.43	66.67
	VP TAG TAG R		25	60.00	71.43
	NX NX NX R		25	12.00	75.00
	SBAR SBAR SBAR R		19	78.95	83.33
	PP PP PP R		14	85.71	63.16
	...				
	TOTAL		763	61.47	62.20

Table 5
Accuracy for various types/sub-types of dependency (part 1). Only sub-types occurring more than 10 times are shown.

Type	Sub-type	Description	Count	Recall	Precision
Mod'n within Base-NPs	NPB TAG TAG L		11786	94.60	93.46
	NPB TAG NPB L		358	97.49	92.82
12742 = 29.6% of all cases	NPB TAG TAG R		189	74.07	75.68
	NPB TAG ADJP L		167	65.27	71.24
	NPB TAG QP L		110	80.91	81.65
	NPB TAG NAC L		29	51.72	71.43
	NPB NX TAG L		27	14.81	66.67
	NPB QP TAG L		15	66.67	76.92
	...				
	TOTAL		12742	93.20	92.59
Mod'n to NPs	NP NPB NP R	Appositive	495	74.34	75.72
	NP NPB SBAR R	Relative clause	476	79.20	79.54
1418 = 3.6% of all cases	NP NPB VP R	Reduced relative	205	77.56	72.60
	NP NPB SG R		63	88.89	81.16
	NP NPB PRN R		53	45.28	60.00
	NP NPB ADVP R		48	35.42	54.84
	NP NPB ADJP R		48	62.50	69.77
	...				
	TOTAL		1418	73.20	75.49
Sentential head	TOP TOP S R		1757	96.36	96.85
	TOP TOP SINV R		89	96.63	94.51
1917 = 4.8% of all cases	TOP TOP NP R		32	78.12	60.98
	TOP TOP SG R		15	40.00	33.33
	...				
	TOTAL		1917	94.99	94.99
Adjunct to a verb	VP TAG ADVP R		367	74.93	78.57
	VP TAG TAG R		349	90.54	93.49
2242 = 5.6% of all cases	VP TAG ADJP R		259	83.78	80.37
	S VP ADVP L		255	90.98	84.67
	VP TAG NP R		187	66.31	74.70
	VP TAG SBAR R		180	74.44	72.43
	VP TAG SG R		159	60.38	68.57
	S VP TAG L		115	86.96	90.91
	S VP SBAR L		81	88.89	85.71
	VP TAG ADVP L		79	51.90	49.40
	S VP PRN L		58	25.86	48.39
	S VP NP L		45	66.67	63.83
	S VP SG L		28	75.00	52.50
	VP TAG PRN R		27	3.70	12.50
	VP TAG S R		11	9.09	100.00
	...				
	TOTAL		2242	75.11	78.44

Table 6
Accuracy for various types/sub-types of dependency (part 2). Only sub-types occurring more than 10 times are shown.

lations of the form ⟨ S VP ** ⟩ where ** is any complement, or ⟨ VP TAG ** ⟩ where ** is any complement except VP-C (i.e., auxiliary-verb—verb dependencies are excluded). The most frequent verb complements, Subject-verb and Object-verb, are recovered with over 95% and 92% precision/recall respectively.

Other complements: 94.47/94.12 recall/precision. This type includes any dependencies where the modifier is a complement, and the dependency does not fall into the *complement to a verb* type.

PP Modification: 82.29/81.51 recall/precision. Any dependency where the modifier is a PP.

Coordination: 61.47/62.20 recall/precision.

Modification within base-NPs: 93.20/92.59 recall/precision. Any dependency where the parent is NPB.

Modification to NPs: 73.20/75.49 recall/precision. Any dependency where the parent is NP, the head is NPB, and the modifier is not a PP.

Sentential Head: 94.99/94.99 recall/precision. Dependency involving the head-word of the entire sentence.

Adjunct to a verb: 75.11/78.44 recall/precision. Any dependency where the parent is VP, the head is TAG, and the modifier is not a PP; or where the parent is S, the head is VP, and the modifier is not a PP.

A conclusion to draw from these accuracies is that the parser is doing very well at recovering the core structure of sentences: complements, sentential heads, and base-NP relationships (NP chunks) are all recovered with over 90% accuracy. The main sources of errors are adjuncts. Coordination is especially difficult, most likely because it often involves a dependency between two content-words, leading to very sparse statistics.

7.2 More about the Distance Measure

The distance measure, whose implementation was described in section 3.1.1, deserves more discussion and motivation. In this section we consider it from three perspectives: its influence on parsing accuracy; an analysis of distributions in training data that are sensitive to the distance variables; and some examples of sentences where it is useful in discriminating between competing analyses.

7.2.1 The Impact of the Distance Measure on Accuracy Table 7 shows the results for models 1 and 2 with and without the adjacency and verb distance measures. It is clear that the distance measure improves accuracy.

What is most striking is just how badly model 1 performs without the distance measure. Looking at the parser's output, the reason for this is that the adjacency condition in the distance measure is approximating subcategorization information. In particular, in phrases such as PPs and SBARs (and, to a lesser extent, in VPs) which almost always take exactly one complement to the right of their head, the adjacency feature encodes this mono-valency through parameters $P(\texttt{STOP}|\texttt{PP/SBAR, adjacent}) = 0$ and $P(\texttt{STOP}|\texttt{PP/SBAR, not adjacent}) = 1$. Figure 14 shows some particularly bad structures returned by model 1 with no distance variables.

The other surprise is that subcategorization can be very useful, but that the distance measure has masked this utility. One interpretation in moving from the least parameterized model Model1(No,No) to the fully parameterized model Model2(Yes,Yes) is that the

MODEL	A	V	LR	LP	CBs	0 CBs	≤ 2 CBs
Model 1	NO	NO	75.0%	76.5%	2.18	38.5%	66.4
Model 1	YES	NO	86.6%	86.7%	1.22	60.9%	81.8
Model 1	YES	YES	87.8%	88.2%	1.03	63.7%	84.4
Model 2	NO	NO	85.1%	86.8%	1.28	58.8%	80.3
Model 2	YES	NO	87.7%	87.8%	1.10	63.8%	83.2
Model 2	YES	YES	88.7%	89.0%	0.95	65.7%	85.6

Table 7
Results on Section 0 of the WSJ Treebank. A = YES, V = YES mean that the adjacency/verb conditions respectively were used in the distance measure. **LR/LP** = labeled recall/precision. **CBs** is the average number of crossing brackets per sentence. **0 CBs,** ≤ 2 **CBs** are the percentage of sentences with 0 or ≤ 2 crossing brackets respectively.

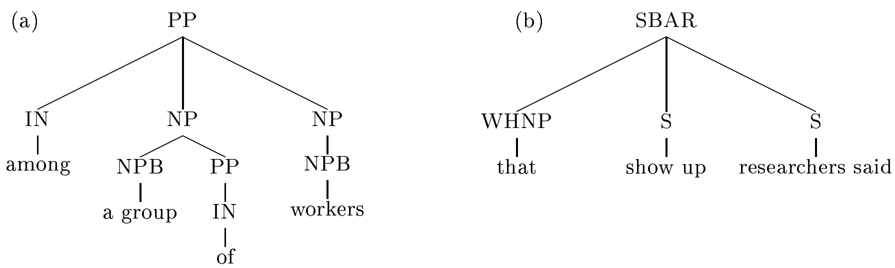
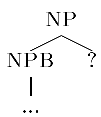


Figure 14
Two examples of bad parses produced by model 1 with no distance or subcategorization conditions (Model1(No,No) in table 7). In (a) one PP has two complements, the other has none; in (b) the SBAR has two complements. In both examples either the adjacency condition or the subcategorization parameters will correct the errors, so these are examples where the adjacency and subcategorization variables overlap in their utility.

adjacency condition adds around 11% in accuracy; the verb condition adds another 1.5%; and subcategorization finally adds a mere 0.8%. Under this interpretation subcategorization information isn't all that useful (and this was my original assumption, as this was the order in which features were originally added to the model). But under another interpretation subcategorization is very useful: in moving from Model1(No,No) to Model2(No,No) we see a 10% improvement due to subcategorization parameters; adjacency then adds a 1.5% improvement; and the verb condition adds a final 1% improvement.

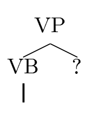
From an engineering point of view, given a choice of whether to add just distance or subcategorization to the model, distance is preferable. But linguistically it is clear that adjacency can only approximate subcategorization, and that subcategorization is more "correct" in some sense. In free word order languages distance may not approximate subcategorization at all well — a complement may appear to either the right or left of the head, confusing the adjacency condition.

7.2.2 Frequencies in Training Data Tables 8 and 9 show the effect of distance on the distribution of modifiers in two of the most frequent syntactic environments: NP and verb modification. The distribution varies a great deal with distance. Most striking is the way that the probability of STOP increases with increasing distance: from 71% to 89% to 98% in the NP case, from 8% to 60% to 96% in the verb case. Each modifier probability generally decreases with distance. For example, the probability of seeing a PP modifier to an NP decreases from 17.7% to 5.57% to 0.93%.



A=TRUE,V=FALSE		A=FALSE,V=FALSE		A=FALSE,V=TRUE	
%age	?	%age	?	%age	?
70.78	STOP	88.53	STOP	97.65	STOP
17.7	PP	5.57	PP	0.93	PP
3.54	SBAR	2.28	SBAR	0.55	SBAR
3.43	NP	1.55	NP	0.35	NP
2.22	VP	0.92	VP	0.22	VP
0.61	SG	0.38	SG	0.09	SG
0.56	ADJP	0.26	PRN	0.07	PRN
0.54	PRN	0.22	ADVP	0.04	ADJP
0.36	ADVP	0.15	ADJP	0.03	ADVP
0.08	TO	0.09	-RRB-	0.02	S
0.08	CONJP	0.02	UCP	0.02	-RRB-
0.03	UCP	0.01	X	0.01	X
0.02	JJ	0.01	RRC	0.01	VBG
0.01	VBN	0.01	RB	0.01	RB
0.01	RRC				
0.01	FRAG				
0.01	CD				
0.01	-LRB-				

Table 8
Distribution of non-terminals generated as post-modifiers to an NP (see tree to the left), at various distances from the head. A=TRUE means the modifier is adjacent to the head, V=TRUE means there is a verb between the head and the modifier. The distributions were calculated from the first 10000 events for each of the three cases in sections 2-21 of the treebank.



A=TRUE,V=FALSE		A=FALSE,V=FALSE		A=FALSE,V=TRUE	
%age	?	%age	?	%age	?
39	NP-C	59.87	STOP	95.92	STOP
15.8	PP	22.7	PP	1.73	PP
8.43	SBAR-C	3.3	NP-C	0.92	SBAR
8.27	STOP	3.16	SG	0.5	NP
5.35	SG-C	2.71	ADVP	0.43	SG
5.19	ADVP	2.65	SBAR	0.16	ADVP
5.1	ADJP	1.5	SBAR-C	0.14	SBAR-C
3.24	S-C	1.47	NP	0.05	NP-C
2.82	RB	1.11	SG-C	0.04	PRN
2.76	NP	0.82	ADJP	0.02	S-C
2.28	PRT	0.2	PRN	0.01	VBN
0.63	SBAR	0.19	PRT	0.01	VB
0.41	SG	0.09	S	0.01	UCP
0.16	VB	0.06	S-C	0.01	SQ
0.1	S	0.06	-RRB-	0.01	S
0.1	PRN	0.03	FRAG	0.01	FRAG
0.08	UCP	0.02	-LRB-	0.01	ADJP
0.04	VBZ	0.01	X	0.01	-RRB-
0.03	VBN	0.01	VBP	0.01	-LRB-
0.03	VBD	0.01	VB		
0.03	FRAG	0.01	UCP		
0.03	-LRB-	0.01	RB		
0.02	VBG	0.01	INTJ		
0.02	SBARQ				
0.02	CONJP				
0.01	X				
0.01	VBP				
0.01	RBR				
0.01	INTJ				
0.01	DT				
0.01	-RRB-				

Table 9
Distribution of non-terminals generated as post-modifiers to a verb within a `VP` (see tree to the left), at various distances from the head. A=TRUE means the modifier is adjacent to the head, V=TRUE means there is a verb between the head and the modifier. The distributions were calculated from the first 10000 events for each of the distributions in sections 2-21. Auxiliary verbs (verbs taking a `VP` complement to their right) were excluded from these statistics.

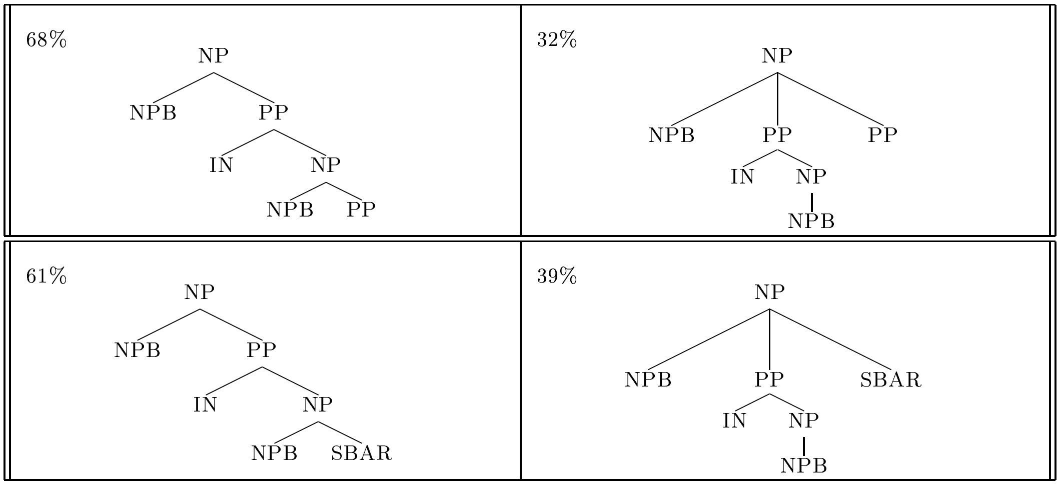


Table 10
Some alternative structures for the same surface sequence of chunks (`NPB PP PP` in the first case, `NPB PP SBAR` in the second case), where the adjacency condition distinguishes between the two structures. The percentages are taken from sections 2-21 of the treebank. In both cases right-branching structures are more frequent.

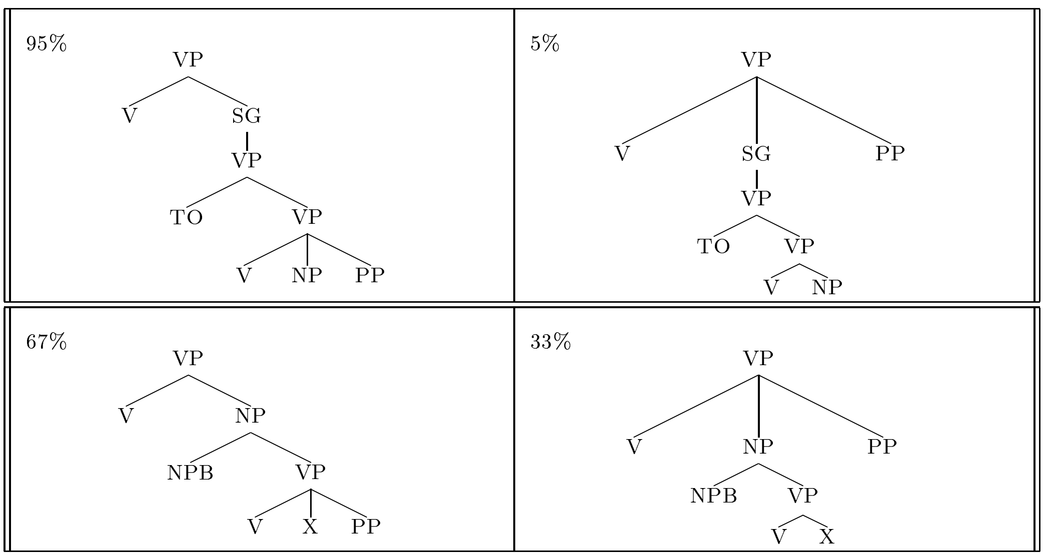


Table 11
Some alternative structures for the same surface sequence of chunks, where the verb condition in the distance measure distinguishes between the two structures. In both cases the low-attachment analyses will get higher probability under the model, due to the low probability of generating a PP modifier involving a dependency that crosses a verb. (X stands for any non-terminal.)

7.2.3 The Distance Features and Right-Branching Structures Both the adjacency and verb components of the distance measure allow the model to learn a preference for right-branching structures. First, consider the adjacency condition. Table 10 shows some examples where right-branching structures are more frequent. Using the statistics from tables 8 and 9, the probability of the alternative structures can be calculated. The results are given below. The right-branching structures get higher probability (although this is before the lexical dependency probabilities are multiplied in, so this "prior" preference for right-branching structures can be over-ruled by lexical preferences). If the distance variables were not conditioned on, the product of terms for the two alternatives would be identical, and the model would have no preference for one structure over another.

Probabilities for the two alternative PP structures in table 10 (excluding probability terms that are constant across the two structures. A=1 means distance is adjacent, A=0 means not adjacent):

Right-branching

$$P(\texttt{PP}|\texttt{NP,NPB,A=1})P(\texttt{STOP}|\texttt{NP,NPB,A=0})$$
$$P(\texttt{PP}|\texttt{NP,NPB,A=1})P(\texttt{STOP}|\texttt{NP,NPB,A=0})$$
$$= 0.177 * 0.8853 * 0.177 * 0.8853 = 0.02455$$

Non Right-branching

$$P(\texttt{PP}|\texttt{NP,NPB,A=1})P(\texttt{PP}|\texttt{NP,NPB,A=0})$$
$$P(\texttt{STOP}|\texttt{NP,NPB,A=0})P(\texttt{STOP}|\texttt{NP,NPB,A=1})$$
$$= 0.177 * 0.0557 * 0.8853 * 0.7078 = 0.006178$$

Probabilities for the SBAR case in table 10, assuming the SBAR contains a verb (V=0 means modification does not cross a verb, V=1 means it does):

Right-branching

$$P(\texttt{PP}|\texttt{NP,NPB,A=1,V=0})P(\texttt{SBAR}|\texttt{NP,NPB,A=1,V=0})$$
$$P(\texttt{STOP}|\texttt{NP,NPB,A=0,V=1})P(\texttt{STOP}|\texttt{NP,NPB,A=0,V=1})$$
$$= 0.177 * 0.0354 * 0.9765 * 0.9765 = 0.005975$$

Non Right-branching

$$P(\texttt{PP}|\texttt{NP,NPB,A=1})P(\texttt{STOP}|\texttt{NP,NPB,A=1})$$
$$P(\texttt{SBAR}|\texttt{NP,NPB,A=0})P(\texttt{STOP}|\texttt{NP,NPB,A=0,V=1})$$
$$= 0.177 * 0.7078 * 0.0228 * 0.9765 = 0.002789$$

7.2.4 The Verb Condition and Right-Branching Structures Table 11 shows some examples where the verb condition is important in differentiating the probability of two structures. In both cases an adjunct can attach either high or low, but the high attachment results in a dependency crossing a verb, and has lower probability.

An alternative to the surface string feature would be a predicate such as "were any of the previous modifiers in X", where X is a set of non-terminals that are likely to contain a verb, such as VP, SBAR, S or SG. This would allow the model to handle cases like the first example in table 11 correctly. The second example shows why it is preferable to condition on the surface string. In this case the verb is "invisible" to the top level, as it is generated recursively below the NP object.

7.2.5 Structural vs. Semantic Preferences One hypothesis would be that lexical statistics are really what is important in parsing: that arriving at a correct interpretation for a sentence is simply a matter of finding the most semantically plausible analysis, and that the statistics related to lexical dependencies approximate this notion of plausibility. Implicitly, we'd be just as well off (maybe even better off) if statistics were calculated between items at the predicate-argument level, with no reference to structure. The distance preferences under this interpretation are just a way of mitigating sparse data problems: when the lexical statistics are too sparse, then falling back on some structural preference is not ideal, but is at least better than chance. This hypothesis is suggested by previous work on specific cases of attachment ambiguity such as PP attachment (e.g., see (Collins and Brooks 95)), which has showed that models will perform better given lexical statistics, and that a straight structural preference is merely a fall-back.

But some examples suggest this is not the case: that, in fact, many sentences have several equally semantically plausible analyses, but that structural preferences distinguish strongly between them. Take the following example (from (Pereira and Warren 80)):

Example 1
John was believed to have been shot by Bill

Surprisingly, this sentence has two analyses — Bill can be the deep subject of either "believed" or "shot". Yet people have a very strong preference for Bill to be doing the shooting, so much so that they may even miss the second analysis. (To see that the dispreferred analysis is semantically quite plausible, consider *Bill believed John to have been shot.*)

As evidence that structural preferences can even over-ride semantic plausibility, take the following example (from (Pinker 1994)):

Example 2
Flip said that Squeaky will do the work yesterday

This sentence is a garden path: the structural preference for "yesterday" to modify the most recent verb is so strong that it is easy to miss the (only) semantically plausible interpretation, paraphrased as *Flip said yesterday that Squeaky will do the work.*

The model makes the correct predictions in these cases. In example 1, the statistics in table 9 show that a PP is 9 times as likely to attach low than high when two verbs are candidate attachment points (the chances of seeing a PP modifier are 15.8% and 1.73% in columns 1 and 3 of the table respectively). In example 2, the probability of seeing an NP (adjunct) modifier to *do* in a non-adjacent but non-verb-crossing environment is 2.11% in sections 2-21 of the treebank (8 out of 379 cases); in contrast the chance of seeing an NP adjunct modifying *said* across a verb is 0.026% (1 out of 3778 cases). The two probabilities differ by a factor of almost 80.

7.3 The Importance of the Choice of Tree Representation

Figures 15 and 16 show some alternative styles of syntactic annotation. The Penn treebank annotation style tends to leave trees quite flat, typically with one level of structure for each X-bar level; at the other extreme are completely binary-branching representations. The two annotation styles are in some sense equivalent, in that it is easy to define a one-to-one mapping between them. But crucially, two different annotation styles may lead to quite different parsing accuracies for a given model, even if the two representations are equivalent under some one-to-one mapping.

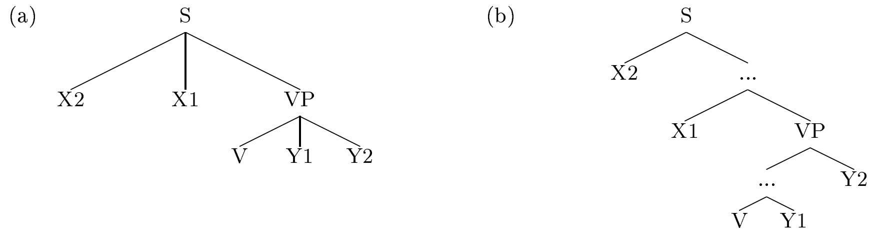


Figure 15
Alternative annotation styles for a sentence S with a verb head V, left modifiers X1, X2, and right modifiers Y1, Y2. (a) is the Penn treebank style of analysis: one level of structure for each bar level. (b) is an alternative but equivalent binary branching representation.

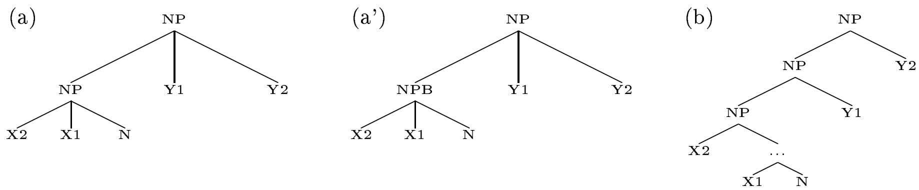


Figure 16
Alternative annotation styles for a noun phrase with a noun head N, left modifiers X1, X2, and right modifiers Y1, Y2. (a) is the Penn treebank style of analysis: one level of structure for each bar level, although notice that the non-recursive as well as recursive noun phrases are labeled NP. (b) is an alternative but equivalent binary branching representation. (a') is our modification of the Penn treebank style to differentiate recursive and non-recursive `NP`s (in some sense `NPB` is a bar 1 structure, `NP` is a bar 2 structure).

A parsing model does not need to be tied to the annotation style of the treebank on which it is trained. The following procedure can be used to transform trees in both training and test data to a new representation:

1. Transform training data trees to the new representation and train the model.
2. Recover parse trees in the new representation when running the model over test data sentences.
3. Convert the test output back to the treebank representation for scoring purposes.

As long as there is a one-to-one mapping between the treebank and the new representation, nothing is lost in doing this. (Goodman 1997) and (Johnson 1997) both suggest this strategy. (Goodman 1997) converts the treebank to binary branching trees. (Johnson 1997) considers conversion to a number of different representations, and discusses how this influences accuracy for non-lexicalized PCFGs.

The models developed in this paper have tacitly assumed the Penn-treebank style of annotation, and will perform badly given other representations (for example, binary-branching trees). This section makes this point more explicit: describing exactly what annotation style is suitable for the models, and showing how other annotation styles will cause problems. This dependence on Penn–treebank style annotations does not imply that the models are inappropriate for a treebank annotated in a different style — in this case we simply recommend transforming the trees to flat, one-level per X-bar level trees before training the model, as in the 3-step procedure outlined above.

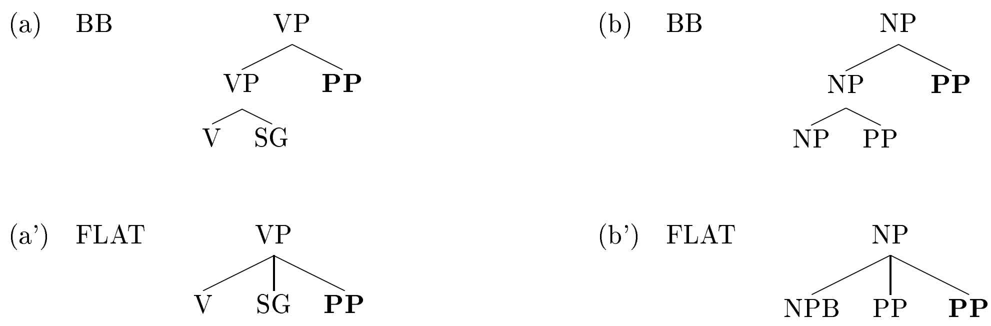


Figure 17
BB = binary branching structures; FLAT = Penn treebank style annotations. In each case the binary branching annotation style prevents the model from learning that these structures should receive low probability due to the long distance dependency associated with the final `PP` (in bold).

Other models in the literature are also very likely to be sensitive to annotation style. (Charniak 1997)'s models will most likely perform quite differently with binary branching trees (for example, his current models will learn that rules such as `VP` $\rightarrow$ `V SG PP` are very rare, but with binary branching structures this context-sensitivity will be lost). The models of (Magerman 1995; Ratnaparkhi 1997) use contextual predicates which would most likely need to be modified given a different annotation style. (Goodman 1997)'s models are the exception, as he already specifies that the treebank should be transformed to his chosen representation, binary branching trees.

7.3.1 Representation Affects Structural, not Lexical, Preferences The alternative representations in figures 15 and 16 have the same lexical dependencies (providing that the binary-branching structures are centered about the head of the phrase, as in the examples). The difference between the representations involves structural preferences such as the right-branching preferences encoded by the distance measure. Applying the models in this paper to treebank analyses which use this type of "head-centered" binary-branching tree will fail to result in a model with a distance measure that correctly encodes a preference for right branching structures.

To see this, consider the examples in figure 17. In each binary branching example the generation of the final modifying `PP` is "blind" to the distance between it and the head that it modifies. At the top level of the tree it is apparently adjacent to the head; crucially the closer modifier (`SG` in (a), the other `PP` in (b)) is hidden lower in the tree structure. So the model will be unable to differentiate generation of the `PP` in adjacent vs. non-adjacent or non-verb-crossing vs. verb-crossing environments, and the structures in figure 17 will get unreasonably high probability.

This does not mean that distance preferences cannot be encoded in a binary branching PCFG. (Goodman 1997) achieves this by adding distance features to the non-terminals. The spirit of this implementation is that the top level rules `VP` $\rightarrow$ `VP PP` and `NP` $\rightarrow$ `NP PP` would be modified to `VP` $\rightarrow$ `VP(+rverb) PP` and `NP` $\rightarrow$ `NP(+rmod) PP`, where (+rverb) means a phrase where the head has a verb in its right-modifiers, and (+rmod) means a phrase that has at least one right-modifier to the head. The model will learn from training data that $P(\texttt{VP} \rightarrow \texttt{VP(+rverb) PP}|\texttt{VP}) \ll P(\texttt{VP} \rightarrow \texttt{VP(-rverb) PP}|\texttt{VP})$, i.e., that a prepositional phrase modification is much more likely when it does not cross a verb.

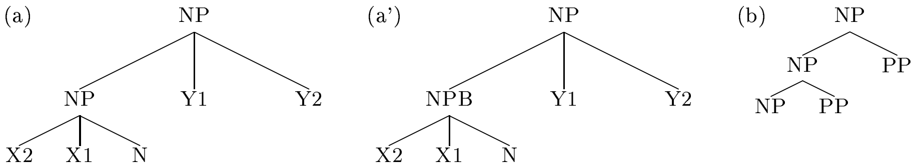


Figure 18
(a) The way the Penn treebank annotates NPs. (a') Our modification to the annotation, to differentiate recursive (NP) vs. non-recursive (NPB) noun phrases. (b) a structure that is never seen in training data, but will receive much too high probability from a model trained on trees of style (a).

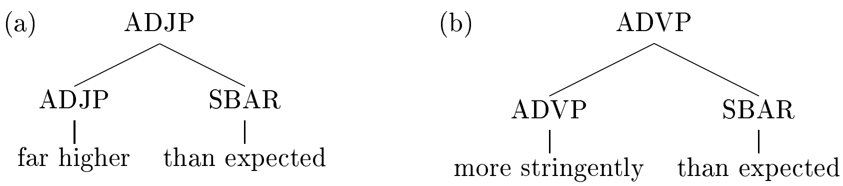


Figure 19
Examples of other phrases in the Penn treebank where non-recursive and recursive phrases are not differentiated.

7.3.2 The Importance of Differentiating Non-recursive vs. Recursive NPs Figure 18 shows the modification to the Penn treebank annotation to relabel base-NPs as NPB. It also illustrates a problem that arises if this distinction is not made: structures such as that in figure 18(b) receive high probability even if they are never seen in training data. ((Johnson 1997) notes that this structure has higher probability than the correct, flat structure, given counts taken from the treebank for a standard PCFG.) The model is fooled by the binary branching style into modeling both PPs as being adjacent to the head of the noun-phrase, so 18(b) will get very high probability.

This problem does not only apply to NPs — other phrases such as adjectival phrases (ADJPs) or adverbial phrases (ADVPs) also have non-recursive (bar 1) and recursive (bar 2) levels, which are not differentiated in the Penn treebank. See figure 19 for examples. Ideally these cases should be differentiated too: we did not implement this change because it is unlikely to make much difference to accuracy given the relative infrequency of these cases (excluding coordination cases, and looking at the 80,254 instances in sections 2-21 of the Penn treebank where a parent and head non-terminal were the same: 94.5% were the NP case; 2.6% were cases of coordination where a punctuation mark was the coordinator[18]; only 2.9% were similar to those in figure 19).

7.3.3 Summary To summarise, the models in this paper assume:

1. Tree representations are "flat": i.e., one level per X-bar level.

2. Different X-bar levels have different labels (in particular, non-recursive vs. recursive levels are differentiated, at least for the most frequent case of NPs).

7.4 The Need to Break Down Rules

The parsing approaches we have described concentrate on breaking down context-free rules in the treebank into smaller components. Lexicalized rules were initially broken

18 for example, (S (S John eats apples) ; (S Mary eats bananas))

down to bare-bones Markov processes, then increased dependency on previously generated modifiers was built back up through the distance measure and subcategorization. Even with this additional context, the models are still able to recover rules in test data that have never been seen in training data.

An alternative, proposed in (Charniak 1997), is to limit parsing to those context free rules seen in training data. A lexicalized rule is predicted in two steps. First, the whole context-free rule is generated. Second, the lexical items are filled in. The probability of a rule is estimated as[19]:

$$P(L_n(l_n)...L_1(l_1)H(h)R_1(r_1)...R_m(r_m) \mid P,h) =$$
$$P(L_n...L_1HR_1...R_m) \mid P,h) \times \prod_{i=1...n} P_l(l_i \mid L_i,P,h) \times \prod_{j=1...m} P_r(r_j \mid R_j,P,h)$$

The estimation technique used in (Charniak 1997) for the CF rule probabilities interpolates several estimates, the lowest being $P(L_n...L_1HR_1...R_m) \mid P)$. Any rules not seen in training data will get zero probability under this model. Parse trees in test data will be limited to include rules seen in training.

A problem with this approach is coverage. As shown in this section, many test data sentences will require rules that have not been seen in training. This gives motivation for breaking down rules into smaller components. This section motivates the need to break down rules from four perspectives. First, we discuss how the Penn treebank annotation style leads to a very large number of grammar rules. Second, we assess the extent of the coverage problem by looking at rule frequencies in training data. Third, we run experiments to assess the impact of the coverage problem on accuracy. Fourth, we discuss how breaking rules down may improve estimation as well as coverage.

7.4.1 The Penn Treebank Annotation Style Leads to Many Rules The "flatness" of the Penn treebank annotation style has already been discussed in section 7.3. The flatness of the trees leads to a very large (and constantly growing) number of rules. The prime reason for this is that the number of adjuncts to a head is potentially unlimited, for example there can be any number of PP adjuncts to a head verb. A binary-branching (Chomsky adjunction) grammar can generate an unlimited number of adjuncts with very few rules. For example, the following grammar generates any sequence VP → V NP PP*:

VP → V NP
VP → VP PP

In contrast, the Penn treebank style would create a new rule for each number of PPs seen in training data. The grammar would be

VP → V NP
VP → V NP PP
VP → V NP PP PP
VP → V NP PP PP PP
.... and so on

Other adverbial adjuncts, such as adverbial phrases or adverbial SBARs, can also modify a verb several times; and all of these different types of adjuncts can be seen together in

19 Charniak's model also conditions on the parent of the non-terminal being expanded, we omit this here for brevity.

the same rule. The result is a combinatorial explosion in the number of rules. To give a flavour of this, here is a random sample of rules that occurred only once in sections 2-21 of the Penn treebank, and were of the format VP $\rightarrow$ VB modifier*:

```
VP → VB NP NP NP PRN
VP → VB NP SBAR PP SG ADVP
VP → VB NP ADVP ADVP PP PP
VP → VB RB
VP → VB NP PP NP SBAR
VP → VB NP PP SBAR PP
```

It is not only verb phrases that cause this kind of combinatorial explosion: other phrases, in particular non-recursive noun phrases, also contribute a huge number of rules. The next section considers the distributional properties of the rules in more detail.

Note that there is good motivation for the Penn treebank's decision to represent rules in this way, rather than with rules expressing Chomsky adjunction (i.e., a schema where complements and adjuncts are separated, through rule types ⟨VP $\rightarrow$ VB {complement}*⟩ and ⟨VP $\rightarrow$ VP {adjunct}⟩). First, it allowed the argument/adjunct distinction for PP modifiers to verbs to be left undefined: this decision was found to be very difficult for annotators. Second, in the *surface* ordering (as opposed to deep structure), adjuncts are often found closer to the head than complements, thereby yielding structures that fall outside the Chomsky adjunction schema. For example, a rule such as ⟨VP $\rightarrow$ VB NP-C PP SBAR-C⟩ is found very frequently in the Penn treebank; SBAR complements nearly always extrapose over adjuncts.

7.4.2 Quantifying the Coverage Problem In order to quantify the coverage problem, rules were collected from sections 2-21 of the Penn treebank. Punctuation was raised as high as possible in the tree, and the rules did not have complement markings or the distinction between base-NPs and recursive NPs. 939,382 rule tokens were collected in this way; there were 12,409 distinct rule types. We also collected the count for each rule. Table 12 shows some statistics for these rules.

A majority of rules in the grammar — 6,765, or 54.5% — occurred only once. These rules account for 0.72% of rules by token. That is, if a rule was drawn at random from the 939,382 rule tokens in sections 2-21 of the treebank, there would be a 0.72% chance of that being the only instance of that rule. On the other hand, when drawing a rule at random from the 12,409 rules in the grammar induced from those sections, there would be a 54.5% chance of that rule having occurred only once.

The percentage by token of the 1-count rules is an indication of the coverage problem. From this estimate, 0.72% of all rules (or 1 in 139 rules) required in test data would never have been seen in training. It was also found that 15.0% (1 in 6.67) of all sentences had at least one rule that occurred just once. This gives an estimate that roughly 1 in 6.67 sentences in test data will not be covered by a grammar induced from 40,000 sentences in the treebank.

If the complement markings are added to the non-terminals, and the base-NP/non-recursive NP distinction is made, then the coverage problem is made worse. Table 13 gives the statistics in this case. 17.1% of all sentences (1 in 5.8 sentences) contained at least one 1-count rule.

7.4.3 The Impact of Coverage on Accuracy Parsing experiments were used to assess the impact of the coverage problem on parsing accuracy. Section 0 of the treebank was parsed with models 1 and 2 as before, but the parse trees were restricted to include rules already seen in training data. Table 14 shows the results. Restricting the rules leads

Rule Count	No. of Rules by Type	Percentage by Type	No. of Rules by token	Percentage by token
1	6765	54.52	6765	0.72
2	1688	13.60	3376	0.36
3	695	5.60	2085	0.22
4	457	3.68	1828	0.19
5	329	2.65	1645	0.18
6 ... 10	835	6.73	6430	0.68
11 ... 20	496	4.00	7219	0.77
21 ... 50	501	4.04	15931	1.70
51 ... 100	204	1.64	14507	1.54
$>$ 100	439	3.54	879596	93.64

Table 12
Statistics for rules taken from sections 2-21 of the treebank, where complement markings were not included on non-terminals.

Rule Count	No. of Rules by Type	Percentage by Type	No. of Rules by token	Percentage by token
1	7865	55.00	7865	0.84
2	1918	13.41	3836	0.41
3	815	5.70	2445	0.26
4	528	3.69	2112	0.22
5	377	2.64	1885	0.20
6 ... 10	928	6.49	7112	0.76
11 ... 20	595	4.16	8748	0.93
21 ... 50	552	3.86	17688	1.88
51 ... 100	240	1.68	16963	1.81
$>$ 100	483	3.38	870728	92.69

Table 13
Statistics for rules taken from sections 2-21 of the treebank, where complement markings were included on non-terminals.

MODEL	Accuracy				
	LR	LP	CBs	0 CBs	$\leq$ 2 CBs
Model 1	87.9	88.3	1.02	63.9	84.4
Model 1 (restricted)	87.4	86.7	1.19	61.7	81.8
Model 2	88.8	89.0	0.94	65.9	85.6
Model 2 (restricted)	87.9	87.0	1.19	62.5	82.4

Table 14
Results on Section 0 of the WSJ Treebank. "restricted" means the model is restricted to recovering rules that have been seen in training data. **LR/LP** = labeled recall/precision. **CBs** is the average number of crossing brackets per sentence. **0 CBs,** ≤ 2 **CBs** are the percentage of sentences with 0 or ≤ 2 crossing brackets respectively.

to a 0.5/1.6% decrease in recall/precision for model 1, and a 0.9/2.0% decrease for model 2.

7.4.4 Breaking Down Rules Improves Estimation Coverage problems are not the only motivation for breaking down rules. The method may also improve estimation. To illustrate this, consider the rules headed by told, whose counts are shown in table 15. Estimating the probability $P(\text{Rule} \mid \text{VP, told})$ using (Charniak 1997)'s method would interpolate two maximum-likelihood estimates:

$$\lambda P_{ml}(\text{Rule} \mid \text{VP, told}) + (1-\lambda)P_{ml}(\text{Rule} \mid \text{VP})$$

Estimation interpolates between the specific, lexically sensitive distribution in table 15, and the non-lexical estimate based on just the parent non-terminal, `VP`. There are many different rules in the more specific distribution (26 different rule types, out of 147 tokens where told was a `VP` head); and there are several 1-count rules (11 cases). From these statistics λ would have to be relatively low. There's a high chance of a new rule for "told" being required in test data, therefore a reasonable amount of probability mass must be left to the backed-off estimate $P_{ml}(\text{Rule} \mid \text{VP})$.

This estimation method is missing a crucial generalisation: in spite of there being many different rules, the distribution over subcategorization frames is much sharper. "told" is seen with only 5 subcategorization frames in training data — the large number of rules is almost entirely due to adjuncts or punctuation appearing after or between complements. The estimation method in model 2 effectively estimates the probability of a rule as

$$P_{lc}(\text{LC} \mid \text{VP, told}) \times P_{rc}(\text{RC} \mid \text{VP, told}) \times P(\text{Rule} \mid \text{VP, told, LC, RC})$$

The left and right subcategorization frames, LC and RC, are chosen first. The entire rule is then generated by Markov processes.

Once armed with the P_{lc} and P_{rc} parameters, the model has the ability to learn the generalisation that "told" appears with a quite limited, sharp distribution over subcategorization frames. Say these parameters are again estimated through interpolation, for example

$$\lambda P_{ml}(\text{LC} \mid \text{VP, told}) + (1-\lambda)P_{ml}(\text{LC} \mid \text{VP})$$

In this case λ can be quite high. Only 5 subcategorization frames (as opposed to 26 rule types) have been seen in the 147 cases. The lexically specific distribution $P_{ml}(\text{LC} \mid \text{VP, told})$ can therefore be quite highly trusted. Relatively little probability mass is left to the backed-off estimate.

In summary, from the distributions in table 15, the model should be quite uncertain about what rules "told" can appear with. However, it should be relatively certain about the subcategorization frame. Introducing subcategorization parameters allows the model to generalise in an important way about rules. We have carefully isolated the "core" of rules — the subcategorization frame — that the model should be certain about.

We should note that Charniak's method will certainly have some advantages in estimation: it will capture some statistical properties of rules that our independence assumptions will lose (e.g., the distribution over the number of `PP` adjuncts seen for a particular head).

8 Related Work

Unfortunately, due to space limitations, it is not possible to give a complete review of previous work in this paper. In the next two sections we give a detailed comparison of

(a)

Count	Rule
70	VP told → VBD NP-C SBAR-C
23	VP told → VBD NP-C
6	VP told → VBD NP-C SG-C
5	VP told → VBD NP-C NP SBAR-C
5	VP told → VBD NP-C : S-C
4	VP told → VBD NP-C PP SBAR-C
4	VP told → VBD NP-C PP
4	VP told → VBD NP-C NP
3	VP told → VBD NP-C PP NP SBAR-C
2	VP told → VBD NP-C PP PP
2	VP told → VBD NP-C NP PP
2	VP told → VBD NP-C , SBAR-C
2	VP told → VBD NP-C , S-C
2	VP told → VBD
2	VP told → ADVP VBD NP-C SBAR-C
1	VP told → VBD NP-C SG-C SBAR
1	VP told → VBD NP-C SBAR-C PP
1	VP told → VBD NP-C SBAR , PP
1	VP told → VBD NP-C PP SG-C
1	VP told → VBD NP-C PP NP
1	VP told → VBD NP-C PP : S-C
1	VP told → VBD NP-C NP : S-C
1	VP told → VBD NP-C ADVP SBAR-C
1	VP told → VBD NP-C ADVP PP NP
1	VP told → VBD NP-C ADVP
1	VP told → VBD NP-C , PRN , SBAR-C
147	TOTAL

(b)

Count	Subcat Frame
89	{NP-C, SBAR-C}
39	{NP-C}
9	{NP-C, S-C}
8	{NP-C, SG-C}
2	{}
147	TOTAL

Table 15
(a) Distribution over rules with “told” as the head (from sections 2-21 of the treebank); (b) Distribution over subcategorization frames with “told” as the head.

the models in this paper to the lexicalized PCFG model of (Charniak 1997), and the history-based models of (Jelinek et al. 1994; Magerman 1995; Ratnaparkhi 1997).

For discussion of additional related work, Chapter 4 of (Collins 1999) attempts to give a comprehensive review of work on statistical parsing up to around 1998. Of particular relevance is other work on parsing the Penn WSJ treebank (Jelinek et al. 1994; Magerman 1995; Eisner 1996; Eisner 1996b; Collins 1996; Charniak 1997; Goodman 1997; Ratnaparkhi 1997; Chelba and Jelinek 1998; Roark 2001). (Eisner 1996; Eisner 1996b) describes several dependency-based models which are also closely related to the models in this paper. (Collins 1996) also describes a dependency-based model applied to treebank parsing. (Goodman 1997) describes probabilistic feature grammars and their application to parsing the treebank. (Chelba and Jelinek 1998) describe an incremental, history-based parsing approach which is applied to language modeling for speech recognition. History-based approaches were introduced to parsing in (Black et al. 1992b). (Roark 2001) describes a generative probabilistic model of an incremental parser, with good results in terms of both parse accuracy on the treebank, and also perplexity scores for language modeling.

Earlier work which is of particular relevance considered the importance of relations between lexical heads for disambiguation in parsing. See (Hindle and Rooth 1991) for one of the earliest pieces of research on this topic, within the context of prepositional phrase attachment ambiguity. For work that used lexical relations for parse disambiguation – all with very promising results – see (Sekine et al 1992; Jones and Eisner 1992a; Jones and Eisner 1992b; Alshawi and Carter 1994). Statistical models of lexicalized grammatical formalisms also lead to models with parameters corresponding to lexical dependencies.

See (Resnik 1992; Schabes 1992; Schabes and Waters 1993) for work on stochastic tree adjoining grammars. (Joshi and Srinivas 1994) describe an alternative, “supertagging” model for tree adjoining grammars. See (Alshawi 1996) for work on stochastic head-automata, and (Lafferty et al. 1992) for a stochastic version of link grammar. (de Marcken 1995) considers stochastic lexicalized PCFGs, with specific reference to EM methods for unsupervised training. (Seneff 1992) describes the use of markov models for rule generation, which is closely related to the markov style rules in the models in the current paper. Finally, note that not all machine learning methods for parsing are probabilistic. See (Brill 1993; Hermjakob and Mooney 1997) for rule-based learning systems.

In recent work, (Chiang 2000) has shown that the models in the current paper can be implemented almost unchanged in a stochastic tree adjoining grammar. (Bikel 2000) has developed generative statistical models which integrate word sense information to the parsing process. (Eisner 2002) develops a sophisticated generative model for lexicalized context-free rules, making use of a probabilistic model of lexicalized transformations between rules. (Blaheta and Charniak 2000) describes methods for the recovery of the semantic tags in the Penn treebank annotations, a significant step on from the complement/adjunct distinction recovered in model 2 of the current paper. (Charniak 2001) gives measurements of perplexity for a lexicalized PCFG. (Gildea 2001) gives experiments investigating the utility of different features in bigram lexical dependency models for parsing. (Miller et al. 2000) develops generative, lexicalized models for information extraction of relations. The model enhances non-terminals in the parse trees to carry semantic labels, and develops a probabilistic model which takes these labels into account. (Collins et al. 1999) describe how the models in the current paper were applied to parsing Czech. (Charniak 2000) describes a parsing model which also uses markov processes to generate rules. The model takes into account much additional context – such as previously generated modifiers, or non-terminals higher in the parse trees – through a maximum-entropy inspired model. These additional features give clear improvements in performance. (Collins 2000) shows similar improvements through a quite different approach based on boosting approaches to reranking (Freund et al., 1998). This approach intends to allow great flexibility in the features which can be incorporated in a model, and additional features are shown to give improvements in parsing performance. Finally, (Bod 2001) describes a very different approach – a DOP approach to parsing – which gives excellent results on treebank parsing, comparable to the results of (Charniak 2000; Collins 2000).

8.1 A Comparison to the Model of Charniak 97

We now give a more detailed comparison of the models in this paper to the parser of (Charniak 1997). The model described in (Charniak 1997) has two types of parameters:

Lexical Dependency Parameters Charniak's dependency parameters are similar to the $L2$ parameters of section 5.1. Whereas our parameters are

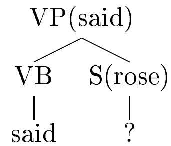
$$P_{L2}(lw_i \mid L_i, lt_i, c, p, P, H, w, t, \Delta, LC)$$

Charniak's parameters in our notation would be

$$P_{L2}(lw_i \mid L_i, P, w)$$

For example, the dependency parameter for an `NP` headed by *profits* being the subject of the verb *rose* would be P(`profits | NP, S, rose`).

Rule Parameters The second type of parameters are associated with context free rules in the tree. As an example, take the `S` node in the following tree:



This non-terminal could expand with any of the rules $S \rightarrow \beta$ in the grammar. The rule probability is defined as $P(S \rightarrow \beta | rose, S, VP)$. So the rule probability depends on the non-terminal being expanded, its headword, and also its parent.

The next few sections give further explanation of the differences between Charniak's models and the models of this thesis.

8.1.1 Additional Features of Charniak's Model There are some notable additional features of Charniak's model. First, the rule probabilities are conditioned on the parent of the non-terminal being expanded. Our models do not include this information, although distinguishing recursive from non-recursive `NP`s can be considered a reduced form of this information. (See section 7.3.2 for a discussion of this distinction; the arguments in that section are also motivation for Charniak's choice of conditioning on the parent.)

Second, Charniak uses word-class information to smooth probabilities, and reports a 0.35% improvement from this feature.

Finally, Charniak uses 30 million words of text for unsupervised training. A parser is trained from the treebank, and used to parse this text; statistics are then collected from this machine-parsed text and merged with the treebank statistics to train a second model. This gave a 0.5% improvement in performance.

8.1.2 The Dependency Parameters of Charniak's Model While similar to ours, Charniak's dependency parameters are conditioned on less information. Whereas our parameters are $P_{L2}(lw_i \mid L_i, lt_i, c, p, P, H, w, t, \Delta, LC)$, Charniak's parameters in our notation would be $P_{L2}(lw_i \mid L_i, P, w)$. The additional information is as follows:

H The head non-terminal label (`VP` in the profits/rose example). At first glance this might seem redundant — for example an `S` will usually take a `VP` as its head. However, in some cases the head label can vary, for example an `S` can take another `S` as its head in coordination cases.

lt_i, t The POS tags for the head and modifier words. This allows our model to use POS tags as word-class information. Charniak's model may be missing an important generalisation in this respect. (Charniak 2000) shows that using the POS tags as word-class information in the model is important for parsing accuracy.

c The coordination flag. This distinguishes, for example, coordination cases from appositives: Charniak's model will have the same parameter — $P(modifier|head, NP, NP)$ — in both of these cases.

p, Δ, **LC/RC** The punctuation, distance and subcategorization variables. It is difficult to tell without empirical tests whether these features are important.

8.1.3 The Rule Parameters of Charniak's Model The rule parameters in Charniak's model are effectively decomposed into our $L1$ parameters (section 5.1), the head parameters, and — in models 2 and 3 — the subcategorization and gap parameters. This decomposition allows our model to assign probability to rules not seen in training data: see section 7.4 for extensive discussion.

8.1.4 Right-Branching Structures in Charniak's Model Our models have used distance features to encode preferences for right-branching structures. Charniak's model does not represent this information explicitly, but instead learns it implicitly through rule probabilities. For example, for an `NP PP PP` sequence the preference for a right-branching structure is encoded through a much higher probability for the rule `NP` $\rightarrow$ `NP PP` rather than `NP` $\rightarrow$ `NP PP PP`. (Notice that conditioning on the rule's parent is needed to disallow the structure `[NP [NP PP] PP]`; see (Johnson 1997) for further discussion.)

This strategy does not encode all of the information in the distance measure. The distance measure effectively penalises rules `NP` $\rightarrow$ `NPB NP PP` where the middle `NP` contains a verb: in this case the `PP` modification results in a dependency that crosses a verb. Charniak's model is unable to distinguish cases where the middle `NP` does/doesn't contain a verb (i.e., the `PP` modification does/doesn't cross a verb).

8.2 A Comparison to The Models of Jelinek et al. 1994, Magerman 1995, Ratnaparkhi 1997

We now make a detailed comparison to the history-based models of (Ratnaparkhi 1997) and (Jelinek et al. 1994; Magerman 1995). A strength of these models is undoubtedly the powerful estimation techniques that they use: maximum entropy modeling (in (Ratnaparkhi 1997)), or decision trees (in (Jelinek et al. 1994; Magerman 1995)). A weakness, we will argue in this section, is the method of associating parameters transitions taken by bottom-up, shift-reduce, style parsers. We give examples where this leads to the parameters unnecessarily fragmenting the training data in some cases, or ignoring important context in other cases. Similar observations have been made in the context of tagging problems using maximum entropy models (Lafferty et al., 2001; Klein and Manning 2002).

We first analyze the model of (Magerman 1995) through three common examples of ambiguity: `PP` attachment, coordination and appositives. In each case a word sequence S has two competing structures — T_1 and T_2 — with associated decision sequences $\langle d_1, ..., d_n \rangle$ and $\langle e_1, ..., e_m \rangle$ respectively. Thus the probability of the two structures can be written as

$$\begin{aligned} P(T_1|S) &= \prod_{i=1...n} P(d_i|d_1...d_{i-1}, S) \\ P(T_2|S) &= \prod_{i=1...m} P(e_i|e_1...e_{i-1}, S) \end{aligned}$$

It will be useful to isolate the decision between the two structures to a single probability term. Let the value j be the minimum value of i such that $d_i \neq e_i$. Then we can rewrite the two probabilities:

$$\begin{aligned} P(T_1|S) &= \prod_{i=1...j-1} P(d_i|d_1...d_{i-1}, S) \times P(d_j|d_1...d_{j-1}, S) \times \prod_{i=j+1...n} P(d_i|d_1...d_{i-1}, S) \\ P(T_2|S) &= \prod_{i=1...j-1} P(e_i|e_1...e_{i-1}, S) \times P(e_j|e_1...e_{j-1}, S) \times \prod_{i=j+1...m} P(e_i|e_1...e_{i-1}, S) \end{aligned}$$

The first thing to note is that $\prod_{i=1...j-1} P(d_i|d_1...d_{i-1}, S) = \prod_{i=1...j-1} P(e_i|e_1...e_{i-1}, S)$, so that these probability terms are irrelevant to the decision between the two structures. We make one additional assumption, that

$$\prod_{i=j+1...n} P(d_i|d_1...d_{i-1}, S) \approx \prod_{i=j+1...m} P(e_i|e_1...e_{i-1}, S) \approx 1$$

This is justified for the examples in this section, because once the jth decision is made, the following decisions are practically deterministic. Equivalently, we are assuming that

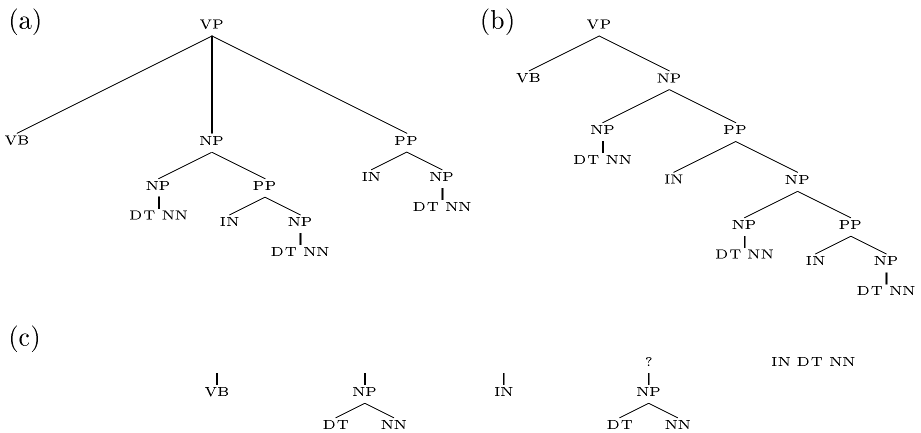


Figure 20
(a) and (b) are two candidate structures for the same sequence of words. (c) shows the first decision (labeled "?") where the two structures differ. The arc above the NP can go either left (for verb attachment of the PP as in (a)) or right (for noun attachment of the PP as in (b)).

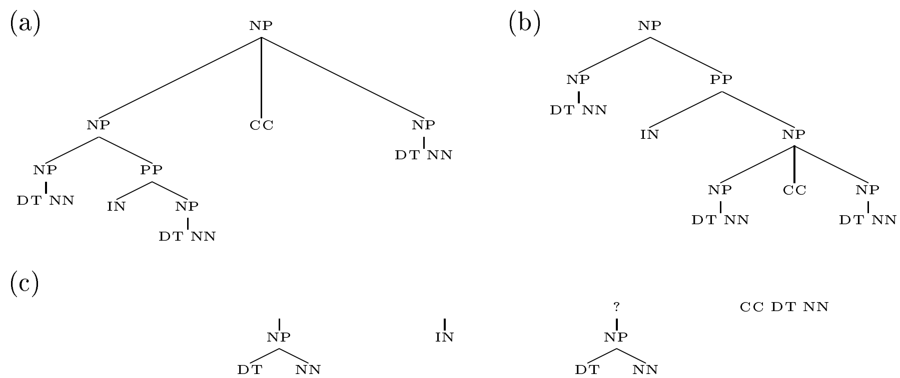


Figure 21
(a) and (b) are two candidate structures for the same sequence of words. (c) shows the first decision (labeled "?") where the two structures differ. The arc above the NP can go either left (for high attachment (a) of the coordinated phrase) or right (for low attachment (b) of the coordinated phrase).

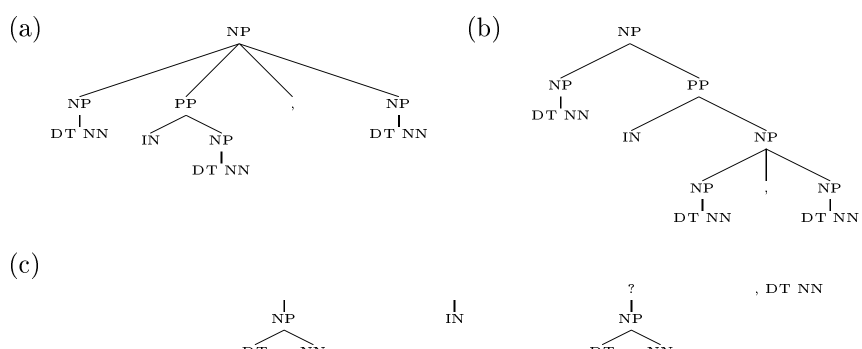


Figure 22
(a) and (b) are two candidate structures for the same sequence of words. (c) shows the first decision (labeled "?") where the two structures differ. The arc above the NP can go either left (for high attachment (a) of the appositive phrase) or right (for noun attachment (b) of the appositive phrase).

$P(T_1|S) + P(T_2|S) \approx 1$, i.e., that very little probability mass is lost to trees other than T_1 or T_2. Given these two equalities, we have isolated the decision between the two structures to the parameters $P(d_j|d_1...d_{j-1}, S)$ and $P(e_j|e_1...e_{j-1}, S)$.

Figure 20 shows a case of PP attachment. The first thing to notice is that the PP attachment decision is made before the PP is even built. The decision is linked to the NP preceding the preposition: whether the arc above the NP should go left or right.

The next thing to notice is that at least one important feature, the verb, falls outside of the conditioning context. (The model only considers information up to two constituents preceding or following the location of the decision.) This could be fixed by considering additional context, but there is no fixed bound on how far the verb can be from the decision point. Note also that in other cases the method fragments the data in unnecessary ways. Cases where the verb directly precedes the NP, or is one place further to the left, are treated separately.

Figure 21 shows a similar example, NP coordination ambiguity. Again, the pivotal decision is made in a somewhat counter-intuitive location: at the NP preceding the coordinator. At this point the NP following the coordinator has not been built, and its head noun is not in the contextual window. Figure 22 shows an appositive example where the head noun of the appositive NP is not in the contextual window when the decision is made.

These last two examples can be extended to illustrate another problem. The NP after the conjunct or comma could be the subject of a following clause. For example, in *John likes Mary and Bill loves Jill* the decision not to coordinate *Mary* and *Bill* is made just after the NP *Mary* is built. At this point, the verb *loves* is outside the contextual window, and the model has no way of telling that *Bill* is the subject of the following clause. The model is assigning probability mass to globally implausible structures due to points of local ambiguity in the parsing process.

Some of these problems can be fixed by changing the derivation order or the conditioning context. (Ratnaparkhi 1997) has an additional chunking stage which means that the head noun does fall within the contextual window for the coordination and appositive cases.

9 Conclusions

The models in this paper incorporate parameters which track a number of linguistic phenomena: bigram lexical dependencies, subcategorization frames, the propagation of slash categories, and so on. The models are generative models, where parse trees are decomposed into a number of steps in a top-down derivation of the tree, and the decisions in the derivation are modeled as conditional probabilities. With a careful choice of derivation and independence assumptions, the resulting model has parameters corresponding to the desired linguistic phenomena.

In addition to introducing the three parsing models, and evaluating their performance on the Penn Wall Street Journal treebank, we have given discussion (in sections 7 and 8) that aims to give more insight into the models: their strengths and weaknesses, the effect of various features on parsing accuracy, and the relationship of the models to other work on statistical parsing. In conclusion, we would like to highlight the following points:

- Section 7.1 showed, through an analysis of accuracy on different types of dependencies, that adjuncts are the main sources of error in the parsing models. In contrast dependencies forming the "core" structure of sentences – for example dependencies involving complements, sentential heads, and NP chunks – are all recovered with over 90% precision and recall.

- Section 7.2 evaluated the effect of the "distance measure" on parsing accuracy. A model without either the adjacency distance feature or subcategorization parameters performs very poorly (76.5/75% precision/recall), suggesting that the adjacency feature is capturing some subcategorization information in the Model 1 parser. The results in table 7 show that the subcategorization, adjacency and "verb-crossing" features all contribute significantly to Model 2's (and by implication Model 3's) performance.
- Section 7.3 described how the three models are well-suited to the Penn-treebank style of annotation, and how certain phenomena (particularly the distance features) may fail to be modeled correctly given treebanks with different annotation styles. This may be an important point to bear in mind when applying the models to other treebanks or other languages. In particular, it may be important to perform transformations on some structures in treebanks with different annotation styles.
- Section 7.4 gave evidence showing the importance of the models' ability to break down the context-free rules in the treebank, thereby generalizing to produce new rules on test examples. Table 14 shows that precision/recall on section 0 of the treebank decreases from 89.0/88.8% to 87.0/87.9% when the model is restricted to produce only those context-free rules seen in training data.
- Section 8 discussed relationships to the generative model of (Charniak 1997), and the history-based (conditional) models of (Ratnaparkhi 1997) and (Jelinek et al. 1994; Magerman 1995). While certainly similar to Charniak's model, the three models in this paper have some significant differences, which are identified in section 8.1. (Another important difference – the ability of Models 1, 2 and 3 to generalize to produce context-free rules not seen in training data – was described in section 7.4.) Section 8.2 showed that the parsing models of (Ratnaparkhi 1997; Jelinek et al. 1994; Magerman 1995) can suffer from very similar problems to the "label bias" or "observation bias" problem observed in tagging models, as described in (Lafferty et al., 2001; Klein and Manning 2002).

Acknowledgments

My PhD thesis is the basis of the work in this paper; I would like to thank Mitch Marcus for being an excellent PhD thesis adviser, and for contributing in many ways to this research. I would like to thank the members of my thesis committee — Aravind Joshi, Mark Liberman, Fernando Pereira and Mark Steedman — for the remarkable breadth and depth of their feedback. The work benefited greatly from discussions with Jason Eisner, Dan Melamed, Adwait Ratnaparkhi, and Paola Merlo. Thanks to Dimitrios Samaras for giving feedback on many portions of the work. I had discussions with many other people at IRCS, University of Pennsylvnia, which contributed quite directly to this research: Breck Baldwin, Srinivas Bangalore, Dan Bikel, James Brooks, Mickey Chandresekhar, David Chiang, Christy Doran, Kyle Hart, Al Kim, Tony Kroch, Robert Macintyre, Max Mintz, Tom Morton, Martha Palmer, Jeff Reynar, Joseph Rosenzweig, Anoop Sarkar, Matthew Stone, Debbie Steinig, Ann Taylor, John Trueswell, Bonnie Webber, Fei Xia, and David Yarowsky. There was also some crucial input from sources outside of Penn. In the summer of 1996 I worked at BBN Technologies: discussions with Scott Miller, Richard Schwartz and Ralph Weischedel had a deep influence on the research. Manny Rayner and David Carter from SRI Cambridge supervised my Masters thesis at Cambridge University: their technical supervision was the beginning of this research. Finally, thanks to the anonymous reviewers for their comments.

References

Abney, Steven. 1997. Stochastic Attribute-Value Grammars. *Computational Linguistics,* 23(4):597-618.

Alshawi, Hiyan. 1996. Head Automata and Bilingual Tiling: Translation with Minimal Representations. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 167-176.

Alshawi, Hiyan and David Carter. Training and Scaling Preference Functions for Disambiguation. *Computational Linguistics,* 20(4):635-648.

Bikel, Dan, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a High-Performance Learning Name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194-201.

Bikel, Dan. 2000. A Statistical Model for Parsing and Word-Sense Disambiguation. In *Proceedings of the Student Research Workshop at ACL 2000.*

Black, Ezra, et al. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop.*

Black, Ezra, Frederick Jelinek, John Lafferty, David Magerman, Robert Mercer and Salim Roukos. 1992. Towards History-Based Grammars: Using Richer Models for Probabilistic Parsing. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*, Harriman, NY.

Blaheta, Don, and Eugene Charniak. 2000. Assigning Function Tags to Parsed Text. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics.* Seattle. pp. 234–240.

Bod, Rens. (2001). What is the Minimal Set of Fragments that Achieves Maximal Parse Accuracy?. In *Proceedings of ACL 2001.*

Booth, Taylor L., and Richard A. Thompson. 1973. *Applying Probability Measures to Abstract Languages.* IEEE Transactions on Computers, C-22(5), pages 442-450.

Brill, Eric. 1993. Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics.*

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, AAAI Press/MIT Press, Menlo Park (1997).

Charniak, Eugene. 2000. *A maximum-entropy-inspired parser.* Proceedings of NAACL 2000.

Charniak, Eugene. 2001. Immediate-Head Parsing for Language Models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics.*

Chelba, Ciprian, and Frederick Jelinek. 1998. Exploiting Syntactic Structure for Language Modeling. In *Proceedings of COLING-ACL 1998*, Montreal.

Chiang, David. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of ACL 2000*, Hong Kong, pages 456-463.

Collins, Michael, and James Brooks. 1995. Prepositional Phrase Attachment through a Backed-off Model. *Proceedings of the Third Workshop on Very Large Corpora*, pages 27-38.

Collins, Michael. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184-191.

Collins, Michael. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16-23.

Collins, Michael. 1999. Head-Driven Statistical Models for Natural Language Parsing. PhD Dissertation, University of Pennsylvania.

Collins, Michael, Jan Hajic, Lance Ramshaw and Christoph Tillmann. 1999. A Statistical Parser for Czech. In *Proceedings of the 37th Annual Meeting of the ACL*, College Park, Maryland.

Collins, Michael. 2000. Discriminative Reranking for Natural Language Parsing. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000).*

Collins, Michael. 2002. Parameter Estimation for Statistical Parsing Models: Theory and Practice of Distribution-Free Methods. To appear as a book chapter.

Eisner, Jason. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. *Proceedings of COLING-96*, pages 340-345.

Eisner, Jason. 1996. An Empirical Comparison of Probability Models for Dependency Grammar. Technical report IRCS-96-11, Institute for Research in Cognitive Science, University of Pennsylvania.

Eisner, Jason, and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the ACL.*

Eisner, Jason. 2002. Transformational priors over grammars. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia.

Freund, Yoav, Raj Iyer, Robert E. Schapire, & Yoram Singer. 1998. An efficient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth International Conference*. Morgan Kaufmann.

Gazdar, Gerald, E.H. Klein, G.K. Pullum, Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Harvard University Press.

Gildea, Daniel. 2001. Corpus Variation and Parser Performance. In *Proceedings of 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Pittsburgh, PA.

Goodman, Joshua. 1997. Probabilistic Feature Grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies.*

Hermjakob, Ulf, and Ray Mooney. Learning Parse and Translation Decisions from Examples with Rich Context. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482-489.

Hindle, Don, and Mats Rooth. 1991. Structural Ambiguity and Lexical Relations. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics.*

Hopcroft, John, and J. D. Ullman. 1979. *Introduction to automata theory, languages, and computation.* Reading, Mass.: Addison-Wesley.

Jelinek, Frederick, John Lafferty, David Magerman, Robert Mercer, Adwait Ratnaparkhi, Salim Roukos. 1994. Decision Tree Parsing using a Hidden Derivation Model. *Proceedings of the 1994 Human Language Technology Workshop*, pages 272-277.

Johnson, Mark. 1997. The Effect of Alternative Tree Representations on Tree Bank Grammars. In *Proceedings of NeMLAP 3.*

Jones, Mark and Jason Eisner. 1992a. A probabilistic parser applied to software testing documents. In *Proceedings of National Conference on Artificial Intelligence (AAAI-92)*, San Jose, pages 322-328.

Jones, Mark and Jason Eisner. 1992b. A probabilistic parser and its application. In *Proceedings of the AAAI-92 Workshop on Statistically-Based Natural Language Processing Techniques*, San Jose.

Joshi, Aravind and Bangalore Srinivas. 1994. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *International Conference on Computational Linguistics (COLING 1994)*, Kyoto University, Japan, August 1994.

Klein, Dan and Christoper Manning. 2002 Conditional Structure versus Conditional Estimation in NLP Models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia.

Lafferty, John, Daniel Sleator and, David Temperley. 1992. Grammatical Trigrams: A Probabilistic Model of Link Grammar. *Proceedings of the 1992 AAAI Fall Symposium on Probabilistic Approaches to Natural Language.*

Lafferty, John, Andrew McCallum, and Fernando Pereira. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 2001.*

Magerman, David. 1995. Statistical Decision-Tree Models for Parsing. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276-283.

Manning, Christopher D., and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing.* MIT Press.

Marcus, Mitchell, Beatrice Santorini and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313-330.

Marcus, Mitchell, Grace Kim, Mary Ann. Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. *Proceedings of the 1994 Human Language Technology Workshop*, pages 110-115.

de Marcken, Carl. 1995. On the Unsupervised Induction of Phrase-Structure Grammars. In *Proceedings of the Third Workshop on Very Large Corpora*.

Miller, Scott, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A Novel Use of Statistical Parsing to Extract Information from Text. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 226-233.

Pereira, Pereira and David Warren. 1980. Definite Clause Grammars for Language Analysis — A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence, 13:231-278.*

Pinker, Stephen. 1994. *The Language Instinct*. Penguin Books.

Ratnaparkhi, Adwait. 1997. A Linear Observed Time Statistical Parser Based on Maximum Entropy Models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, Brown University, Providence, Rhode Island.

Ratnaparkhi, Adwait. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. *Conference on Empirical Methods in Natural Language Processing*, May 1996.

Resnik, Philip. 1992. Probabilistic Tree-Adjoining Grammar as a Framework for Statistical Natural Language Processing. In *Proceedings of COLING 1992*, Volume II, pages 418–424.

Roark, Brian. 2001. Probabilistic top-down parsing and language modeling. In Computational Linguistics, 27(2), pages 249-276.

Schabes, Yves. 1992. Stochastic Lexicalized Tree-Adjoining Grammars. In *Proceedings of COLING 1992*, Volume II, pages 426–432.

Schabes, Yves and Richard Waters. 1993. Stochastic Lexicalized Context-Free Grammar. In *Proceedings of the Third International Workshop on Parsing Technologies.*

Sekine, Satoshi, John Carroll, S. Ananiadou, and J. Tsujii. 1992. Automatic Learning for Semantic Collocation. In *Proceedings of the Third Conference on Applied Natural Language Processing.*

Seneff, Stephanie. 1992. TINA: A Natural Language System for Spoken Language Applications. *Computational Linguistics*, 18(1):61-86.

Witten, Ian and Timothy C. Bell. 1991. The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, July 1991.