# Accurate Unlexicalized Parsing

**Dan Klein**
Computer Science Department
Stanford University
Stanford, CA 94305-9040
klein@cs.stanford.edu

**Christopher D. Manning**
Computer Science Department
Stanford University
Stanford, CA 94305-9040
manning@cs.stanford.edu

## Abstract

We demonstrate that an unlexicalized PCFG can parse much more accurately than previously shown, by making use of simple, linguistically motivated state splits, which break down false independence assumptions latent in a vanilla treebank grammar. Indeed, its performance of 86.36% (LP/LR $F_1$) is better than that of early *lexicalized* PCFG models, and surprisingly close to the current state-of-the-art. This result has potential uses beyond establishing a strong lower bound on the maximum possible accuracy of unlexicalized models: an unlexicalized PCFG is much more compact, easier to replicate, and easier to interpret than more complex lexical models, and the parsing algorithms are simpler, more widely understood, of lower asymptotic complexity, and easier to optimize.

In the early 1990s, as probabilistic methods swept NLP, parsing work revived the investigation of probabilistic context-free grammars (PCFGs) (Booth and Thomson, 1973; Baker, 1979). However, early results on the utility of PCFGs for parse disambiguation and language modeling were somewhat disappointing. A conviction arose that *lexicalized* PCFGs (where head words annotate phrasal nodes) were the key tool for high performance PCFG parsing. This approach was congruent with the great success of word $n$-gram models in speech recognition, and drew strength from a broader interest in lexicalized grammars, as well as demonstrations that lexical dependencies were a key tool for resolving ambiguities such as PP attachments (Ford et al., 1982; Hindle and Rooth, 1993). In the following decade, great success in terms of parse disambiguation and even language modeling was achieved by various lexicalized PCFG models (Magerman, 1995; Charniak, 1997; Collins, 1999; Charniak, 2000; Charniak, 2001).

However, several results have brought into question how large a role lexicalization plays in such parsers. Johnson (1998) showed that the performance of an *unlexicalized* PCFG over the Penn treebank could be improved enormously simply by annotating each node by its parent category. The Penn treebank covering PCFG is a poor tool for parsing because the context-freedom assumptions it embodies are far too strong, and weakening them in this way makes the model much better. More recently, Gildea (2001) discusses how taking the *bilexical* probabilities out of a good current lexicalized PCFG parser hurts performance hardly at all: by at most 0.5% for test text from the same domain as the training data, and not at all for test text from a different domain.[1] But it is precisely these bilexical dependencies that backed the intuition that lexicalized PCFGs should be very successful, for example in Hindle and Rooth's demonstration from PP attachment. We take this as a reflection of the fundamental sparseness of the lexical dependency information available in the Penn Treebank. As a speech person would say, one million words of training data just isn't enough. Even for topics central to the treebank's *Wall Street Journal* text, such as stocks, many very plausible dependencies occur only once, for example *stocks stabilized*, while many others occur not at all, for example *stocks skyrocketed*.[2]

The best-performing lexicalized PCFGs have increasingly made use of *subcategorization*[3] of the

[1]There are minor differences, but all the current best-known lexicalized PCFGs employ both *monolexical* statistics, which describe the phrasal categories of arguments and adjuncts that appear around a head lexical item, and *bilexical* statistics, or dependencies, which describe the likelihood of a head word taking as a dependent a phrase headed by a certain other word.

[2]This observation motivates various class- or similarity-based approaches to combating sparseness, and this remains a promising avenue of work, but success in this area has proven somewhat elusive, and, at any rate, current lexicalized PCFGs do simply use exact word matches if available, and interpolate with syntactic category-based estimates when they are not.

[3]In this paper we use the term *subcategorization* in the original general sense of Chomsky (1965), for where a syntactic cat-

categories appearing in the Penn treebank. Charniak (2000) shows the value his parser gains from parent-annotation of nodes, suggesting that this information is at least partly complementary to information derivable from lexicalization, and Collins (1999) uses a range of linguistically motivated and carefully hand-engineered subcategorizations to break down wrong context-freedom assumptions of the naive Penn treebank covering PCFG, such as differentiating "base NPs" from noun phrases with phrasal modifiers, and distinguishing sentences with empty subjects from those where there is an overt subject NP. While he gives incomplete experimental results as to their efficacy, we can assume that these features were incorporated because of beneficial effects on parsing that were complementary to lexicalization.

In this paper, we show that the parsing performance that can be achieved by an unlexicalized PCFG is far higher than has previously been demonstrated, and is, indeed, much higher than community wisdom has thought possible. We describe several simple, linguistically motivated annotations which do much to close the gap between a vanilla PCFG and state-of-the-art lexicalized models. Specifically, we construct an *unlexicalized* PCFG which outperforms the *lexicalized* PCFGs of Magerman (1995) and Collins (1996) (though not more recent models, such as Charniak (1997) or Collins (1999)).

One benefit of this result is a much-strengthened lower bound on the capacity of an unlexicalized PCFG. To the extent that no such strong baseline has been provided, the community has tended to greatly overestimate the beneficial effect of lexicalization in probabilistic parsing, rather than looking critically at where lexicalized probabilities are both *needed* to make the right decision and *available* in the training data. Secondly, this result affirms the value of linguistic analysis for feature discovery. The result has other uses and advantages: an unlexicalized PCFG is easier to interpret, reason about, and improve than the more complex lexicalized models. The grammar representation is much more compact, no longer requiring large structures that store lexicalized probabilities. The parsing algorithms have lower asymptotic complexity[4] and have much smaller grammar constants. An unlexicalized PCFG parser is much simpler to build and optimize, including both standard code optimization techniques and the investigation of methods for search space pruning (Caraballo and Charniak, 1998; Charniak et al., 1998).

It is *not* our goal to argue against the use of lexicalized probabilities in high-performance probabilistic parsing. It has been comprehensively demonstrated that lexical dependencies are useful in resolving major classes of sentence ambiguities, and a parser should make use of such information where possible. We focus here on using unlexicalized, structural context because we feel that this information has been underexploited and underappreciated. We see this investigation as only one part of the foundation for state-of-the-art parsing which employs *both* lexical and structural conditioning.

## 1 Experimental Setup

To facilitate comparison with previous work, we trained our models on sections 2–21 of the WSJ section of the Penn treebank. We used the first 20 files (393 sentences) of section 22 as a development set (*devset*). This set is small enough that there is noticeable variance in individual results, but it allowed rapid search for good features via continually reparsing the devset in a partially manual hill-climb. All of section 23 was used as a test set for the final model. For each model, input trees were annotated or transformed in some way, as in Johnson (1998). Given a set of transformed trees, we viewed the local trees as grammar rewrite rules in the standard way, and used (unsmoothed) maximum-likelihood estimates for rule probabilities.[5] To parse the grammar, we used a simple array-based Java implementation of a generalized CKY parser, which, for our final best model, was able to exhaustively parse all sentences in section 23 in 1GB of memory, taking approximately 3 sec for average length sentences.[6]

egory is divided into several subcategories, for example dividing verb phrases into finite and non-finite verb phrases, rather than in the modern restricted usage where the term refers only to the syntactic argument frames of predicators.

[4] $O(n^3)$ vs. $O(n^5)$ for a naive implementation, or vs. $O(n^4)$ if using the clever approach of Eisner and Satta (1999).

[5] The tagging probabilities were smoothed to accommodate unknown words. The quantity $P(tag|word)$ was estimated as follows: words were split into one of several categories $wordclass$, based on capitalization, suffix, digit, and other character features. For each of these categories, we took the maximum-likelihood estimate of $P(tag|wordclass)$. This distribution was used as a prior against which observed taggings, if any, were taken, giving $P(tag|word) = [c(tag, word) + \kappa P(tag|wordclass)]/[c(word)+\kappa]$. This was then inverted to give $P(word|tag)$. The quality of this tagging model impacts all numbers; for example the raw treebank grammar's devset $F_1$ is 72.62 with it and 72.09 without it.

[6] The parser is available for download as open source at: http://nlp.stanford.edu/downloads/lex-parser.shtml

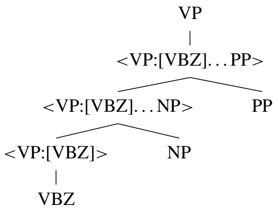



Figure 1: The $v$=1, $h$=1 markovization of VP → VBZ NP PP.

| Vertical Order | | Horizontal Markov Order $h = 0$ | $h = 1$ | $h \leq 2$ | $h = 2$ | $h = \infty$ |
|---|---|---|---|---|---|---|
| $v = 1$ | No annotation | 71.27 | 72.5 | 73.46 | 72.96 | 72.62 |
| | | (854) | (3119) | (3863) | (6207) | (9657) |
| $v \leq 2$ | Sel. Parents | 74.75 | 77.42 | 77.77 | 77.50 | 76.91 |
| | | (2285) | (6564) | (7619) | (11398) | (14247) |
| $v = 2$ | All Parents | 74.68 | 77.42 | 77.81 | 77.50 | 76.81 |
| | | (2984) | (7312) | (8367) | (12132) | (14666) |
| $v \leq 3$ | Sel. GParents | 76.50 | 78.59 | 79.07 | 78.97 | 78.54 |
| | | (4943) | (12374) | (13627) | (19545) | (20123) |
| $v = 3$ | All GParents | 76.74 | 79.18 | 79.74 | 79.07 | 78.72 |
| | | (7797) | (15740) | (16994) | (22886) | (22002) |

Figure 2: Markovizations: $F_1$ and grammar size.

## 2 Vertical and Horizontal Markovization

The traditional starting point for unlexicalized parsing is the raw $n$-ary treebank grammar read from training trees (after removing functional tags and null elements). This basic grammar is imperfect in two well-known ways. First, the category symbols are too coarse to adequately render the expansions independent of the contexts. For example, subject NP expansions are very different from object NP expansions: a subject NP is 8.7 times more likely than an object NP to expand as just a pronoun. Having separate symbols for subject and object NPs allows this variation to be captured and used to improve parse scoring. One way of capturing this kind of external context is to use *parent annotation*, as presented in Johnson (1998). For example, NPs with S parents (like subjects) will be marked NP^S, while NPs with VP parents (like objects) will be NP^VP.

The second basic deficiency is that many rule types have been seen only once (and therefore have their probabilities overestimated), and many rules which occur in test sentences will never have been seen in training (and therefore have their probabilities underestimated – see Collins (1999) for analysis). Note that in parsing with the unsplit grammar, not having seen a rule doesn't mean one gets a parse failure, but rather a possibly very weird parse (Charniak, 1996). One successful method of combating sparsity is to *markovize* the rules (Collins, 1999). In particular, we follow that work in markovizing out from the head child, despite the grammar being unlexicalized, because this seems the best way to capture the traditional linguistic insight that phrases are organized around a head (Radford, 1988).

Both parent annotation (adding context) and RHS markovization (removing it) can be seen as two instances of the same idea. In parsing, every node has a vertical history, including the node itself, parent, grandparent, and so on. A reasonable assumption is that only the past $v$ vertical ancestors matter to the current expansion. Similarly, only the previous $h$ horizontal ancestors matter (we assume that the head child always matters). It is a historical accident that the default notion of a treebank PCFG grammar takes $v = 1$ (only the current node matters vertically) and $h = \infty$ (rule right hand sides do not decompose at all). On this view, it is unsurprising that increasing $v$ and decreasing $h$ have historically helped.

As an example, consider the case of $v = 1$, $h = 1$. If we start with the rule VP → VBZ NP PP PP, it will be broken into several stages, each a binary or unary rule, which conceptually represent a head-outward generation of the right hand size, as shown in figure 1. The bottom layer will be a unary over the head declaring the goal: $\langle$VP: [VBZ]$\rangle$ → VBZ. The square brackets indicate that the VBZ is the head, while the angle brackets $\langle$X$\rangle$ indicates that the symbol $\langle$X$\rangle$ is an intermediate symbol (equivalently, an active or incomplete state). The next layer up will generate the first rightward sibling of the head child: $\langle$VP: [VBZ]... NP$\rangle$ → $\langle$VP: [VBZ]$\rangle$ NP. Next, the PP is generated: $\langle$VP: [VBZ]... PP$\rangle$ → $\langle$VP: [VBZ]... NP$\rangle$ PP. We would then branch off left siblings if there were any.[7] Finally, we have another unary to finish the VP. Note that while it is convenient to think of this as a head-outward process, these are just PCFG rewrites, and so the actual scores attached to each rule will correspond to a downward generation order.

Figure 2 presents a grid of horizontal and vertical markovizations of the grammar. The raw treebank grammar corresponds to $v = 1, h = \infty$ (the upper right corner), while the parent annotation in (Johnson, 1998) corresponds to $v = 2, h = \infty$, and the second-order model in Collins (1999), is broadly a smoothed version of $v = 2, h = 2$. In addition to exact $n$th-order models, we tried variable-

[7] In our system, the last few right children carry over as preceding context for the left children, distinct from common practice. We found this wrapped horizon to be beneficial, and it also unifies the infinite order model with the unmarkovized raw rules.

| | Cumulative | | | Indiv. |
|---|---|---|---|---|
| Annotation | Size | $F_1$ | $\Delta$ $F_1$ | $\Delta$ $F_1$ |
| Baseline ($v \leq 2, h \leq 2$) | 7619 | 77.77 | – | – |
| UNARY-INTERNAL | 8065 | 78.32 | 0.55 | 0.55 |
| UNARY-DT | 8066 | 78.48 | 0.71 | 0.17 |
| UNARY-RB | 8069 | 78.86 | 1.09 | 0.43 |
| TAG-PA | 8520 | 80.62 | 2.85 | 2.52 |
| SPLIT-IN | 8541 | 81.19 | 3.42 | 2.12 |
| SPLIT-AUX | 9034 | 81.66 | 3.89 | 0.57 |
| SPLIT-CC | 9190 | 81.69 | 3.92 | 0.12 |
| SPLIT-% | 9255 | 81.81 | 4.04 | 0.15 |
| TMP-NP | 9594 | 82.25 | 4.48 | 1.07 |
| GAPPED-S | 9741 | 82.28 | 4.51 | 0.17 |
| POSS-NP | 9820 | 83.06 | 5.29 | 0.28 |
| SPLIT-VP | 10499 | 85.72 | 7.95 | 1.36 |
| BASE-NP | 11660 | 86.04 | 8.27 | 0.73 |
| DOMINATES-V | 14097 | 86.91 | 9.14 | 1.42 |
| RIGHT-REC-NP | 15276 | 87.04 | 9.27 | 1.94 |

Figure 3: Size and devset performance of the cumulatively annotated models, starting with the markovized baseline. The right two columns show the change in $F_1$ from the baseline for each annotation introduced, both cumulatively and for each single annotation applied to the baseline in isolation.

history models similar in intent to those described in Ron et al. (1994). For variable horizontal histories, we did not split intermediate states below 10 occurrences of a symbol. For example, if the symbol ⟨VP: [VBZ]... PP PP⟩ were too rare, we would collapse it to ⟨VP: [VBZ]... PP⟩. For vertical histories, we used a cutoff which included both frequency and mutual information between the history and the expansions (this was not appropriate for the horizontal case because MI is unreliable at such low counts).

Figure 2 shows parsing accuracies as well as the number of symbols in each markovization. These symbol counts include all the intermediate states which represent partially completed constituents. The general trend is that, in the absence of further annotation, more vertical annotation is better – even exhaustive grandparent annotation. This is not true for horizontal markovization, where the variable-order second-order model was superior. The best entry, $v = 3$, $h \leq 2$, has an $F_1$ of 79.74, already a substantial improvement over the baseline.

In the remaining sections, we discuss other annotations which increasingly split the symbol space. Since we expressly do not smooth the grammar, not all splits are guaranteed to be beneficial, and not all sets of useful splits are guaranteed to co-exist well. In particular, while $v = 3$, $h \leq 2$ markovization is good on its own, it has a large number of states and does not tolerate further splitting well. Therefore, we base all further exploration on the $v \leq 2, h \leq 2$

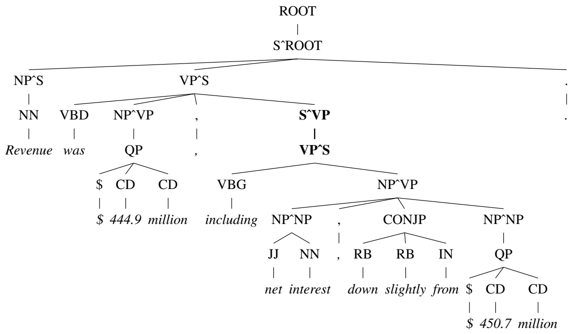



Figure 4: An error which can be resolved with the UNARY-INTERNAL annotation (incorrect baseline parse shown).

grammar. Although it does not necessarily jump out of the grid at first glance, this point represents the best compromise between a compact grammar and useful markov histories.

## 3 External vs. Internal Annotation

The two major previous annotation strategies, parent annotation and head lexicalization, can be seen as instances of external and internal annotation, respectively. Parent annotation lets us indicate an important feature of the external environment of a node which influences the internal expansion of that node. On the other hand, lexicalization is a (radical) method of marking a distinctive aspect of the otherwise hidden internal contents of a node which influence the external distribution. Both kinds of annotation can be useful. To identify split states, we add suffixes of the form -X to mark internal content features, and ^X to mark external features.

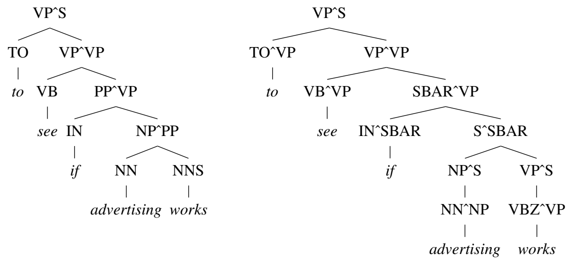
To illustrate the difference, consider unary productions. In the raw grammar, there are many unaries, and once any major category is constructed over a span, most others become constructible as well using unary chains (see Klein and Manning (2001) for discussion). Such chains are rare in real treebank trees: unary rewrites only appear in very specific contexts, for example S complements of verbs where the S has an empty, controlled subject. Figure 4 shows an erroneous output of the parser, using the baseline markovized grammar. Intuitively, there are several reasons this parse should be ruled out, but one is that the lower S slot, which is intended primarily for S complements of communication verbs, is not a unary rewrite position (such complements usually have subjects). It would therefore be natural to annotate the trees so as to confine unary productions to the contexts in which they are actually appropriate. We tried two annotations. First, UNARY-

INTERNAL marks (with a -U) any nonterminal node which has only one child. In isolation, this resulted in an absolute gain of 0.55% (see figure 3). The same sentence, parsed using only the baseline and UNARY-INTERNAL, is parsed correctly, because the VP rewrite in the incorrect parse ends with an SˆVP-U with very low probability.[8]

Alternately, UNARY-EXTERNAL, marked nodes which had no siblings with ˆU. It was similar to UNARY-INTERNAL in solo benefit (0.01% worse), but provided far less marginal benefit on top of other later features (none at all on top of UNARY-INTERNAL for our top models), and was discarded.[9] One restricted place where external unary annotation was very useful, however, was at the preterminal level, where internal annotation was meaningless. One distributionally salient tag conflation in the Penn treebank is the identification of demonstratives (*that, those*) and regular determiners (*the, a*). Splitting DT tags based on whether they were only children (UNARY-DT) captured this distinction. The same external unary annotation was even more effective when applied to adverbs (UNARY-RB), distinguishing, for example, *as well* from *also*). Beyond these cases, unary tag marking was detrimental. The $F_1$ after UNARY-INTERNAL, UNARY-DT, and UNARY-RB was 78.86%.

## 4 Tag Splitting

The idea that part-of-speech tags are not fine-grained enough to abstract away from specific-word behaviour is a cornerstone of lexicalization. The UNARY-DT annotation, for example, showed that the determiners which occur alone are usefully distinguished from those which occur with other nominal material. This marks the DT nodes with a single bit about their immediate external context: whether there are sisters. Given the success of parent annotation for nonterminals, it makes sense to parent annotate tags, as well (TAG-PA). In fact, as figure 3 shows, exhaustively marking all preterminals with their parent category was the most effective single annotation we tried. Why should this be useful? Most tags have a canonical category. For example, NNS tags occur under NP nodes (only 234 of 70855 do not, mostly mistakes). However, when a tag




(a) (b)

Figure 5: An error resolved with the TAG-PA annotation (of the IN tag): (a) the incorrect baseline parse and (b) the correct TAG-PA parse. SPLIT-IN also resolves this error.

somewhat regularly occurs in a non-canonical position, its distribution is usually distinct. For example, the most common adverbs directly under ADVP are *also* (1599) and *now* (544). Under VP, they are *n't* (3779) and *not* (922). Under NP, *only* (215) and *just* (132), and so on. TAG-PA brought $F_1$ up substantially, to 80.62%.

In addition to the adverb case, the Penn tag set conflates various grammatical distinctions that are commonly made in traditional and generative grammar, and from which a parser could hope to get useful information. For example, subordinating conjunctions (*while, as, if*), complementizers (*that, for*), and prepositions (*of, in, from*) all get the tag IN. Many of these distinctions are captured by TAG-PA (subordinating conjunctions occur under S and prepositions under PP), but are not (both subordinating conjunctions and complementizers appear under SBAR). Also, there are exclusively noun-modifying prepositions (*of*), predominantly verb-modifying ones (*as*), and so on. The annotation SPLIT-IN does a linguistically motivated 6-way split of the IN tag, and brought the total to 81.19%.

Figure 5 shows an example error in the baseline which is equally well fixed by either TAG-PA or SPLIT-IN. In this case, the more common nominal use of *works* is preferred unless the IN tag is annotated to allow *if* to prefer S complements.

We also got value from three other annotations which subcategorized tags for specific lexemes. First we split off auxiliary verbs with the SPLIT-AUX annotation, which appends ˆBE to all forms of *be* and ˆHAVE to all forms of *have*.[10] More minorly, SPLIT-CC marked conjunction tags to indicate

[8] Note that when we show such trees, we generally only show one annotation on top of the baseline at a time. Moreover, we do not explicitly show the binarization implicit by the horizontal markovization.

[9] These two are not equivalent even given infinite data.

[10] This is an extended uniform version of the partial auxiliary annotation of Charniak (1997), wherein all auxiliaries are marked as AUX and a -G is added to gerund auxiliaries and gerund VPs.

whether or not they were the strings [*Bb*]*ut* or &, each of which have distinctly different distributions from other conjunctions. Finally, we gave the percent sign (%) its own tag, in line with the dollar sign ($) already having its own. Together these three annotations brought the $F_1$ to 81.81%.

## 5 What is an Unlexicalized Grammar?

Around this point, we must address exactly what we mean by an *unlexicalized* PCFG. To the extent that we go about subcategorizing POS categories, many of them might come to represent a single word. One might thus feel that the approach of this paper is to walk down a slippery slope, and that we are merely arguing degrees. However, we believe that there is a fundamental qualitative distinction, grounded in linguistic practice, between what we see as permitted in an unlexicalized PCFG as against what one finds and hopes to exploit in lexicalized PCFGs. The division rests on the traditional distinction between *function words* (or closed-class words) and *content words* (or open class or lexical words). It is standard practice in linguistics, dating back decades, to annotate phrasal nodes with important function-word distinctions, for example to have a CP[*for*] or a PP[*to*], whereas content words are not part of grammatical structure, and one would not have special rules or constraints for an NP[*stocks*], for example. We follow this approach in our model: various closed classes are subcategorized to better represent important distinctions, and important features commonly expressed by function words are annotated onto phrasal nodes (such as whether a VP is finite, or a participle, or an infinitive clause). However, no use is made of lexical class words, to provide either monolexical or bilexical probabilities.[11]

At any rate, we have kept ourselves honest by estimating our models exclusively by maximum likelihood estimation over our subcategorized grammar, without any form of interpolation or shrinkage to unsubcategorized categories (although we do markovize rules, as explained above). This effectively means that the subcategories that we break off must themselves be very frequent in the language. In such a framework, if we try to annotate categories with any detailed lexical information, many sentences either entirely fail to parse, or have only extremely weird parses. The resulting battle against sparsity means that we can only afford to make a few distinctions which have major distributional impact. Even with the individual-lexeme annotations in this section, the grammar still has only 9255 states compared to the 7619 of the baseline model.

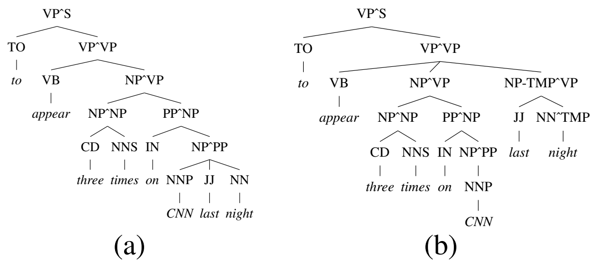


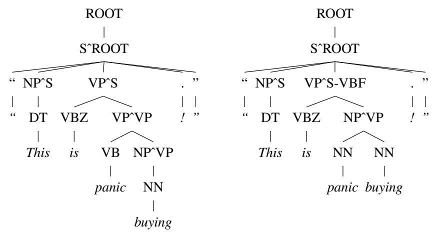

Figure 6: An error resolved with the TMP-NP annotation: (a) the incorrect baseline parse and (b) the correct TMP-NP parse.

## 6 Annotations Already in the Treebank

At this point, one might wonder as to the wisdom of stripping off all treebank functional tags, only to heuristically add other such markings back in to the grammar. By and large, the treebank out-of-the package tags, such as PP-LOC or ADVP-TMP, have negative utility. Recall that the raw treebank grammar, with no annotation or markovization, had an $F_1$ of 72.62% on our development set. With the functional annotation left in, this drops to 71.49%. The $h \le 2, v \le 1$ markovization baseline of 77.77% dropped even further, all the way to 72.87%, when these annotations were included.

Nonetheless, some distinctions present in the raw treebank trees were valuable. For example, an NP with an S parent could be either a temporal NP or a subject. For the annotation TMP-NP, we retained the original -TMP tags on NPs, and, furthermore, propagated the tag down to the tag of the head of the NP. This is illustrated in figure 6, which also shows an example of its utility, clarifying that *CNN last night* is not a plausible compound and facilitating the otherwise unusual high attachment of the smaller NP. TMP-NP brought the cumulative $F_1$ to 82.25%. Note that this technique of pushing the functional tags down to preterminals might be useful more generally; for example, locative PPs expand roughly the

[11] It should be noted that we started with four tags in the Penn treebank tagset that rewrite as a single word: EX (*there*), WP$ (*whose*), # (the pound sign), and TO), and some others such as WP, POS, and some of the punctuation tags, which rewrite as barely more. To the extent that we subcategorize tags, there will be more such cases, but many of them already exist in other tag sets. For instance, many tag sets, such as the Brown and CLAWS (c5) tagsets give a separate sets of tags to each form of the verbal auxiliaries *be*, *do*, and *have*, most of which rewrite as only a single word (and any corresponding contractions).




(a) (b)

Figure 7: An error resolved with the SPLIT-VP annotation: (a) the incorrect baseline parse and (b) the correct SPLIT-VP parse.

same way as all other PPs (usually as IN NP), but they do tend to have different prepositions below IN.

A second kind of information in the original trees is the presence of empty elements. Following Collins (1999), the annotation GAPPED-S marks S nodes which have an empty subject (i.e., raising and control constructions). This brought $F_1$ to 82.28%.

## 7 Head Annotation

The notion that the head word of a constituent can affect its behavior is a useful one. However, often the head tag is as good (or better) an indicator of how a constituent will behave.[12] We found several head annotations to be particularly effective. First, possessive NPs have a very different distribution than other NPs – in particular, NP $\rightarrow$ NP $\alpha$ rules are only used in the treebank when the leftmost child is possessive (as opposed to other imaginable uses like for *New York lawyers*, which is left flat). To address this, POSS-NP marked all possessive NPs. This brought the total $F_1$ to 83.06%. Second, the VP symbol is very overloaded in the Penn treebank, most severely in that there is no distinction between finite and infinitival VPs. An example of the damage this conflation can do is given in figure 7, where one needs to capture the fact that present-tense verbs do not generally take bare infinitive VP complements. To allow the finite/non-finite distinction, and other verb type distinctions, SPLIT-VP annotated all VP nodes with their head tag, merging all finite forms to a single tag VBF. In particular, this also accomplished Charniak's gerund-VP marking. This was extremely useful, bringing the cumulative $F_1$ to 85.72%, 2.66% absolute improvement (more than its solo improvement over the baseline).

[12] This is part of the explanation of why (Charniak, 2000) finds that early generation of head tags as in (Collins, 1999) is so beneficial. The rest of the benefit is presumably in the availability of the tags for smoothing purposes.

## 8 Distance

Error analysis at this point suggested that many remaining errors were attachment level and conjunction scope. While these kinds of errors are undoubtedly profitable targets for lexical preference, most attachment mistakes were overly high attachments, indicating that the overall right-branching tendency of English was not being captured. Indeed, this tendency is a difficult trend to capture in a PCFG because often the high and low attachments involve the very same rules. Even if not, attachment height is not modeled by a PCFG unless it is somehow explicitly encoded into category labels. More complex parsing models have indirectly overcome this by modeling distance (rather than height).

Linear distance is difficult to encode in a PCFG – marking nodes with the size of their yields massively multiplies the state space.[13] Therefore, we wish to find indirect indicators that distinguish high attachments from low ones. In the case of two PPs following a NP, with the question of whether the second PP is a second modifier of the leftmost NP or should attach lower, inside the first PP, the important distinction is usually that the lower site is a non-recursive base NP. Collins (1999) captures this notion by introducing the notion of a base NP, in which any NP which dominates only preterminals is marked with a -B. Further, if an NP-B does not have a non-base NP parent, it is given one with a unary production. This was helpful, but substantially less effective than marking base NPs *without* introducing the unary, whose presence actually erased a useful internal indicator – base NPs are more frequent in subject position than object position, for example. In isolation, the Collins method actually hurt the baseline (absolute cost to $F_1$ of 0.37%), while skipping the unary insertion added an absolute 0.73% to the baseline, and brought the cumulative $F_1$ to 86.04%.

In the case of attachment of a PP to an NP either above or inside a relative clause, the high NP is distinct from the low one in that the already modified one contains a verb (and the low one may be a base NP as well). This is a partial explanation of the utility of verbal distance in Collins (1999). To

[13] The inability to encode distance naturally in a naive PCFG is somewhat ironic. In the heart of any PCFG parser, the fundamental table entry or chart item is a label over a span, for example an NP from position 0 to position 5. The concrete use of a grammar rule is to take two adjacent span-marked labels and combine them (for example NP[0,5] and VP[5,12] into S[0,12]). Yet, only the labels are used to score the combination.

| Length $\leq$ 40 | LP | LR | $F_1$ | Exact | CB | 0 CB |
|---|---|---|---|---|---|---|
| Magerman (1995) | 84.9 | 84.6 | | | 1.26 | 56.6 |
| Collins (1996) | 86.3 | 85.8 | | | 1.14 | 59.9 |
| this paper | 86.9 | 85.7 | 86.3 | 30.9 | 1.10 | 60.3 |
| Charniak (1997) | 87.4 | 87.5 | | | 1.00 | 62.1 |
| Collins (1999) | 88.7 | 88.6 | | | 0.90 | 67.1 |

| Length $\leq$ 100 | LP | LR | $F_1$ | Exact | CB | 0 CB |
|---|---|---|---|---|---|---|
| this paper | 86.3 | 85.1 | 85.7 | 28.8 | 1.31 | 57.2 |

Figure 8: Results of the final model on the test set (section 23).

capture this, DOMINATES-V marks all nodes which dominate any verbal node (V*, MD) with a -V. This brought the cumulative $F_1$ to 86.91%. We also tried marking nodes which dominated prepositions and/or conjunctions, but these features did not help the cumulative hill-climb.

The final distance/depth feature we used was an explicit attempt to model depth, rather than use distance and linear intervention as a proxy. With RIGHT-REC-NP, we marked all NPs which contained another NP on their right periphery (i.e., as a rightmost descendant). This captured some further attachment trends, and brought us to a final development $F_1$ of 87.04%.

## 9 Final Results

We took the final model and used it to parse section 23 of the treebank. Figure 8 shows the results. The test set $F_1$ is 86.32% for $\leq$ 40 words, already higher than early lexicalized models, though of course lower than the state-of-the-art parsers.

## 10 Conclusion

The advantages of unlexicalized grammars are clear enough – easy to estimate, easy to parse with, and time- and space-efficient. However, the dismal performance of basic unannotated unlexicalized grammars has generally rendered those advantages irrelevant. Here, we have shown that, surprisingly, the maximum-likelihood estimate of a compact unlexicalized PCFG can parse on par with early lexicalized parsers. We do not want to argue that lexical selection is not a worthwhile component of a state-of-the-art parser – certain attachments, at least, require it – though perhaps its necessity has been overstated. Rather, we have shown ways to improve parsing, some easier than lexicalization, and others of which are orthogonal to it, and could presumably be used to benefit lexicalized parsers as well.

## Acknowledgements

This paper is based on work supported in part by the National Science Foundation under Grant No. IIS-0085896, and in part by an IBM Faculty Partnership Award to the second author.

## References

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

Taylor L. Booth and Richard A. Thomson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22:442–450.

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24:275–298.

Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 127–133.

Eugene Charniak. 1996. Tree-bank grammars. In *Proc. of the 13th National Conference on Artificial Intelligence*, pp. 1031–1036.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pp. 598–603.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *NAACL 1*, pages 132–139.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *ACL 39*.

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

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *ACL 34*, pages 184–191.

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

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *ACL 37*, pages 457–464.

Marilyn Ford, Joan Bresnan, and Ronald M. Kaplan. 1982. A competence-based theory of syntactic closure. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 727–796. MIT Press, Cambridge, MA.

Daniel Gildea. 2001. Corpus variation and parser performance. In *2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:613–632.

Dan Klein and Christopher D. Manning. 2001. Parsing with treebank grammars: Empirical bounds, theoretical models, and the structure of the Penn treebank. In *ACL 39/EACL 10*.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *ACL 33*, pages 276–283.

Andrew Radford. 1988. *Transformational Grammar*. Cambridge University Press, Cambridge.

Dana Ron, Yoram Singer, and Naftali Tishby. 1994. The power of amnesia. *Advances in Neural Information Processing Systems*, volume 6, pages 176–183. Morgan Kaufmann.