In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (AAAI-99)

The Role of Lexicalization and Pruning for Base Noun Phrase Grammars

Claire Cardie and **David Pierce**
Department of Computer Science
Cornell University
Ithaca, NY 14853
cardie, pierce@cs.cornell.edu

Abstract

This paper explores the role of lexicalization and pruning of grammars for base noun phrase identification. We modify our original framework (Cardie & Pierce 1998) to extract lexicalized treebank grammars that assign a score to each potential noun phrase based upon both the part-of-speech tag sequence and the word sequence of the phrase. We evaluate the modified framework on the "simple" and "complex" base NP corpora of the original study. As expected, we find that lexicalization dramatically improves the performance of the unpruned treebank grammars; however, for the simple base noun phrase data set, the lexicalized grammar performs below the corresponding unlexicalized but pruned grammar, suggesting that lexicalization is not critical for recognizing very simple, relatively unambiguous constituents. Somewhat surprisingly, we also find that error-driven pruning improves the performance of the probabilistic, lexicalized base noun phrase grammars by up to 1.0% recall and 0.4% precision, and does so even using the original pruning strategy that fails to distinguish the effects of lexicalization. This result may have implications for many probabilistic grammar-based approaches to problems in natural language processing: error-driven pruning is a remarkably robust method for improving the performance of probabilistic and non-probabilistic grammars alike.

Introduction

Base noun phrase identification (see Figure 1) is a critical component in many large-scale natural language processing (NLP) applications: it is among the first steps for many partial parsers; information retrieval systems rely on base noun phrases as a primary source of linguistic phrases for indexing; base noun phrases support information extraction, a variety of text-mining operations, and distributional clustering techniques that attempt to relieve sparse data problems. As a result, a number of researchers have targeted the problem of base noun phrase recognition (Church 1988; Bourigault 1992; Voutilainen 1993; Justeson & Katz 1995).

[The survival] of [spinoff Cray Computer Corp.]
as [a fledgling] in [the supercomputer business]
appears to depend heavily on [the creativity] —
and [longevity] — of [its chairman] and
[chief designer], [Seymour Cray].

Base noun phrases: simple, nonrecursive noun phrases — noun phrases that do not contain other noun phrase descendants.

Figure 1: Base NP Examples

Only recently, however, have efforts in this area attempted the automatic acquisition of base noun phrase (base NP) parsers and their automatic evaluation on the same large test corpus: Ramshaw & Marcus (1998) applied transformation-based learning (Brill 1995); Argamon, Dagan, & Krymolowski (1998) devised a memory-based sequence learning (MBSL) method; previously we introduced error-driven pruning of treebank grammars (Cardie & Pierce 1998). All three methods for base NP recognition have been evaluated using part-of-speech tagged and base NP annotated corpora derived from the Penn Treebank (Marcus, Marcinkiewicz, & Santorini 1993), thus offering opportunity for more direct comparison than algorithms that have been evaluated by hand.

Ramshaw & Marcus's noun phrase bracketer learns a set of transformation rules. Each transformation locally updates the noun phrase bracketing associated with a single word based on nearby features, such as neighboring words, part-of-speech tags, and bracket boundaries. After the training phase, the learned transformations are applied, in order, to each novel text to identify base NPs. Argamon *et al.* develop a variation of memory-based learning (Stanfill & Waltz 1986) for base NP recognition. During training, their MBSL algorithm saves the entire raw training corpus. Generalization of the implicit noun phrase rules in the training corpus occurs at application time — MBSL searches the novel text for tag sequences or combinations of tag subsequences (tiles) that occurred during training in a similar context.

Our corpus-based algorithm for noun phrase recognition uses a simpler representation for the base NP grammar, namely part-of-speech tag sequences. It extracts the tag sequence grammar from the treebank training corpus, then prunes it using an error-based benefit metric. To identify a base NP in a novel sentence, a simple longest-match bracketer scans input text from left to right, at each point selecting the longest sequence of tags matching a grammar rule (if any) to form a base NP. The approach has a number of attractive features: both the training and the bracketer are very simple; the bracketer is very fast; the learned grammar can be easily modified. Nonetheless, while the accuracy of the treebank approach is very good for applications that require or prefer fairly simple base NPs, it lags the alternative approaches when identifying more complex noun phrases. This can be explained in part by examining the sources of knowledge employed by each method: The treebank approach uses neither lexical (i.e. word-based) information nor context; MBSL captures context in its tiles, but uses no lexical information; the transformation-based learner uses both lexical information and the surrounding context to make decisions about bracket boundary placement. Context and lexicalization have been shown to be important across a variety of natural language learning tasks; as a result, we might expect the treebank approach to improve with the addition of either. However, lexicalization and its accompanying transition to a probabilistic grammar has at least one goal similar to that of pruning: to reduce the effect of "noisy" rules in the grammar. Therefore, it is not clear that lexicalization alone will improve the error-driven pruning treebank approach to base noun phrase recognition.

This paper explores the role of lexicalization and pruning of base noun phrase grammars. More specifically, we modify our original framework to extract lexicalized treebank grammars that assign a score to each potential noun phrase based upon both the tag sequence and the word sequence of the phrase. In addition, we extend the noun phrase bracketer to select the combination of brackets with the highest score. We evaluate the modified framework on the "simple" and "complex" base NP corpora of the original study.

As expected, we find that lexicalization dramatically improves the performance of the unpruned treebank grammars, with an increase in precision and recall of approximately 70% and 50%, respectively. However, for the simple base NP data set, the lexicalized grammar still performs below the unlexicalized, but pruned, grammar of the original base NP study, suggesting that lexicalization is not critical for recognizing very simple, relatively unambiguous constituents. For the simple base NP task, pruning serves much the same function as lexicalization in that both suppress the application of bad rules. Pruning, however, allows the simplicity of the grammar and bracketing procedure to remain intact. In contrast, for more complex base NPs, the lexicalized grammar performs comparably to the pruned unlexicalized grammar of the original study. Thus, the importance of lexical information appears to increase with the complexity of the linguistic task: more than just the tag sequence is needed to determine the quality of a candidate phrase when the targets are more ambiguous. For many applications, however, the added complexity of the lexicalized approach may not be worth the slight increase in performance.

There were a couple of surprises in our results: both the longest-match heuristic of the original bracketer and the original error-driven pruning strategy improved the performance of the lexicalized base NP grammars. First, the longest-match heuristic proved to be more useful than lexical information for the unpruned grammar on both corpora. Second, we found that pruning improves bracketing by up to 1.0% recall and 0.4% precision, and does so even using the original strategy that fails to distinguish the effects of lexicalized rules. This result may have implications for many probabilistic grammar-based approaches to problems in natural language processing: error-driven pruning is a remarkably robust method for improving the performance of probabilistic and non-probabilistic grammars alike.

The next section of this paper defines base noun phrases and reviews the basic framework used to extract, prune, and apply grammars using a treebank base NP corpus. The following section extends the framework to include lexicalized grammars. We then evaluate the modified framework and conclude with a discussion and comparison of approaches to base noun phrase identification.

The Treebank Approach to Base Noun Phrase Identification

In this work we define *base NPs* to be simple, nonrecursive noun phrases — noun phrases that do not contain other noun phrase descendants. The bracketed portions of Figure 1, for example, show the base NPs in one sentence from the Penn Treebank Wall Street Journal corpus. Thus, the string *the survival of spinoff Cray Computer Corp. as a fledgling in the supercomputer business* is too complex to be a base NP; instead, it contains four simpler noun phrases, each of which is considered a base NP: *the survival*, *spinoff Cray Computer Corp.*, *a fledgling*, and *the supercomputer business*. This section reviews the treebank approach to base noun phrase identification depicted in Figure 2. For more detail, see Cardie & Pierce (1998).

Grammar Extraction

Grammar extraction requires a corpus that has been annotated with base NPs. More specifically, we assume that the training corpus is a sequence of words $w_1, w_2, \ldots$, along with a set of base NP annotations $b_{(i_1,j_1)}, b_{(i_2,j_2)}, \ldots$, where $b_{(i,j)}$ indicates that the noun phrase includes words i through j: $[_{\text{NP}}\ w_i, \ldots, w_j]$. The goal of the training phase is to create a base NP grammar from this training corpus:

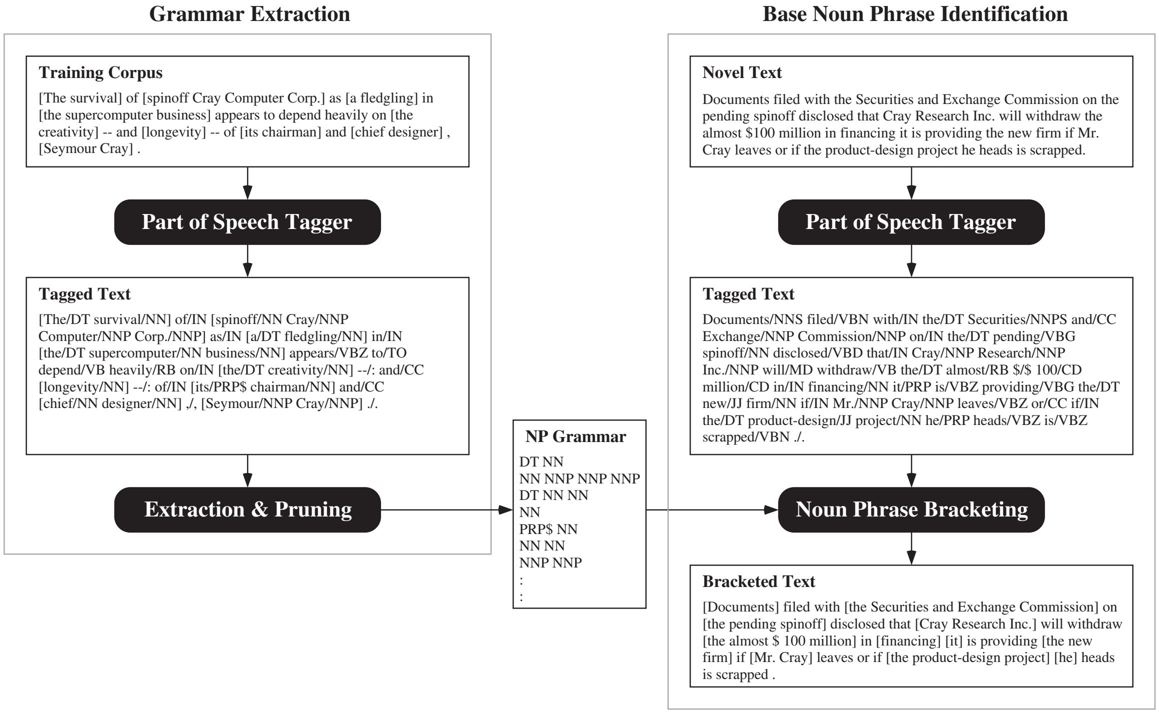

Figure 2: The Treebank Approach to Base NP Identification

1. Using any available part-of-speech tagger, assign a part-of-speech tag t_i to each word w_i in the training corpus.
2. Extract from each base noun phrase $b_{(i,j)}$ in the training corpus its sequence of part-of-speech tags $t_i, \ldots, t_j$ to form base NP rules, one rule per base NP.
3. Remove any duplicate rules.

The resulting grammar can then be used to identify base NPs in a novel text using a "longest-match" heuristic:

1. Assign part-of-speech tags $t_1, t_2, \ldots$ to the input words $w_1, w_2, \ldots$
2. Proceed through the tagged text from left to right, at each point matching the NP rules against the remaining part-of-speech tags $t_i, t_{i+1}, \ldots$ in the text.
3. If there are multiple rules that match beginning at t_i, use the longest matching rule R. Add the new base noun phrase $b_{(i,i+|R|-1)}$ to the set of base NPs. Continue matching at $t_{i+|R|}$.

Unfortunately, these extracted grammars pick up many "bad" rules due to noise in the training data (including annotation errors, part-of-speech tagging errors, and genuine ambiguities). The framework therefore includes a pruning phase whose goal is to eliminate bad rules from the grammar.

Error-Driven Pruning

At a high level, the pruning phase estimates the accuracy of each rule in the grammar using an unseen base NP corpus (the *pruning corpus*); it then eliminates rules with unacceptable performance. This process is illustrated in Figure 3. Like the training corpus, the pruning corpus is annotated with base noun phrases. Initially ignoring the NP annotations, the pruning algorithm first applies the bracketing procedure to the pruning corpus to identify its NPs. The proposed NPs are then compared to the original NPs. Performance of the rule set is measured in terms of labeled precision (P):

$$P = \frac{\text{\# of correct proposed NPs}}{\text{\# of proposed NPs}}$$

We then assign to each rule a score that denotes the "net benefit" achieved by the rule during parsing of the pruning corpus. The benefit of rule r is given by

$$B_r = C_r - E_r$$

where C_r is the number of NPs correctly identified by r, and E_r is the number of precision errors for which r is responsible. This benefit measure is identical to that used in transformation-based learning (Brill 1995) to select an ordered set of useful transformations. The benefit scores from evaluation on the pruning corpus are used to rank the rules in the grammar. With such a ranking, we can improve the rule set by discarding the worst rules. The pruning procedure is then repeated on the resulting rule set.

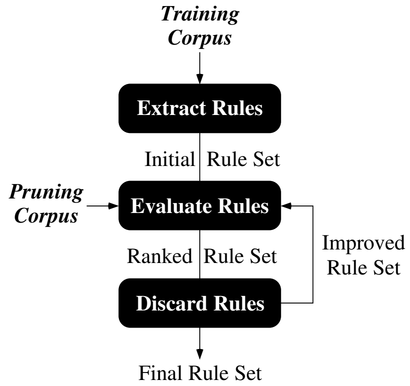

Figure 3: Pruning the Base NP Grammar

We have investigated two iterative approaches for discarding rules, a *thresholding* approach and an *incremental* approach. At each pruning iteration, thresholding discards rules whose score is less than a predefined threshold T. For all of our experiments, we set $T = 1$ to select rules that propose more correct bracketings than incorrect. Threshold pruning repeats until no rules have a score less than T. For our evaluation, this typically requires only four to five iterations. Incremental pruning is a more fine-grained method of discarding rules. At each iteration, incremental pruning discards the N worst rules. In all of our experiments, we set $N = 10$. Incremental pruning repeats until it finds the rule set that maximizes precision on the pruning corpus.

Lexicalization

As noted in the introduction, lexicalized representations have been shown to uniformly increase the accuracy of automated approaches to a variety of natural language learning tasks including syntactic parsing (e.g. Collins (1996), Charniak (1997)), part-of-speech tagging (e.g. Brill (1995)), named entity identification (e.g. Bikel *et al.* (1997)), and word-sense disambiguation (e.g. Ng & Lee (1996)). Ramshaw & Marcus (1998), for example, find that lexicalization accounts for a 2.7% increase in recall and precision for base noun phrase identification. As a result, we believe that lexicalization will also improve the accuracy of the base NP bracketer described above. This section describes lexicalization of the treebank approach to base noun phrase identification.

At a high level, lexicalization uses information about the words in each potential noun phrase bracket to decide how reasonable the bracket is. In our framework, the most straightforward means for incorporating lexical information is to extract a word-based grammar from the training corpus rather than a tag-based grammar. The word-level rules, however, exhibit severe sparse data problems: matching on an exact sequence of words is too strict. Instead, we propose to first find potential brackets using the tag-based grammar, and then use lexical information to further inform the final bracket choices.

More specifically, the goal of the lexicalized bracketer is to use lexical information to assign to each candidate NP bracket b a score $S(b)$, meant to loosely indicate the bracket's likelihood of occurrence.[1] The set of candidate brackets with their scores can be viewed as a weighted graph. Figure 4 shows such a graph (but without arc weights) for the short sentence *Stock prices increased 5 percent yesterday morning.*

- Vertices denote inter-word boundaries. Since the sentence contains seven words, the graph contains eight vertices.
- Weighted "bracket" arcs denote potential NP brackets. For example, *stock* is contained in two bracket arcs: one for the bracket [*stock*] from vertex 0 to 1 and another for [*stock prices*] from 0 to 2.
- Weighted "nonbracket" arcs denote individual words that are not part of any noun phrase bracket. For example, the word *stock* also has a nonbracket arc *stock/NN* from 0 to 1. This allows for the possibility that the word is better off in no NP bracket at all.

In this bracket graph, any path from the source to the sink represents a base NP bracketing of the sentence. For the best combination of brackets, we find the path having the maximum product of arc weights.

Note that the role of pruning for the lexicalized grammars is unclear: both pruning and lexicalization are attempts to increase the performance of a tag sequence grammar. The addition of probabilities to rules — the norm for most lexicalized grammars — is generally believed to preclude the need for pruning rules from the grammar. We investigate the role of pruning for the lexicalized grammars in the Evaluation section.

In the paragraphs below, we provide the remaining details for lexicalizing the treebank approach to base noun phrase identification: the scoring functions used to weight bracket and non-bracket arcs; parameter selection for the lexicalization scoring; and the algorithm for finding the best path through the graph.

Computing a Score for Base NP Bracket Arcs. To score a potential base NP bracket, we consider both the tag sequence r (the rule), and the word sequence p (the phrase), that comprise the bracket. Further, we assume that the first and last words, u and v, of a bracketed phrase are the most informative; thus we drop other words and use (u, v) to approximate p.[2] The score $S(b)$ of a bracket is then a linear combination of several scoring functions that employ frequencies derived from the training corpus. First, there is a score

[1] In future work it may be appropriate to formulate a true probabilistic model of the likelihood that a candidate b is or is not an actual NP; in this work we consider only the much simpler scoring function presented here.

[2] In fact, this simplified instance representation $b = (r, u, v)$ performs comparably to one that considers all the words in a phrase as a set of modifiers M with head h.

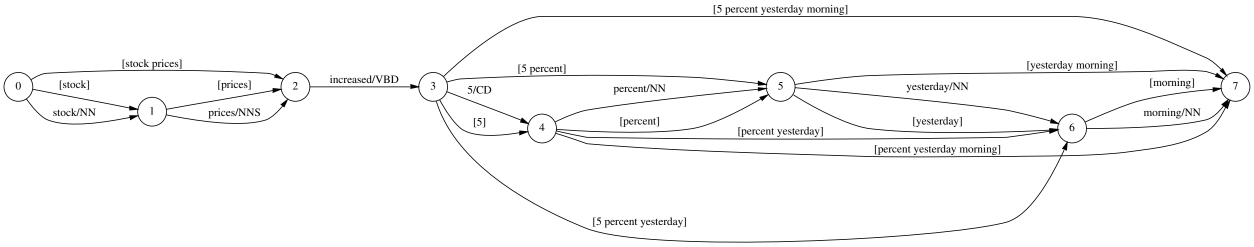

Figure 4: Possible Bracketings of *Stock prices increased 5 percent yesterday morning*

for the rule r.

$$S(r) = \frac{\text{\# of base NPs with tag sequence } r}{\text{\# of occurrences of tag sequence } r}$$

Second, there is a score for the words u, v.

$$S_r(u,v) = \frac{\begin{array}{c}\text{\# of base NPs with tag sequence } r,\\ \text{first word } u \text{, and second word } v\end{array}}{\text{\# of base NPs with tag sequence } r}$$

Since there are not very many training occurrences of brackets with the same rule, first, and last words, we combine $S_r(u,v)$ with similarly computed scores $S_r(u)$ and $S_r(v)$, considering u and v each with r individually.

Last, motivated by the surprising success of the longest-match heuristic, we add a length score $|r|/R$, where R is the maximum rule length. The complete score function is as follows.

$$S(b) = \lambda_r S(r) + \lambda_p[S_r(u,v) + S_r(u) + S_r(v)] + \lambda_l \frac{|r|}{R}$$

Note that the three parameters allow us to determine the right contribution for each source of information. The parameters are constrained to sum to 1.

Computing a Score for Non-Bracket Arcs. The score of a nonbracket arc n is a similar combination of scores for the arc's tag t and word w.

$$S(n) = \lambda_t S(t) + \lambda_w S_t(w)$$

where the tag and word scores are as follows.

$$S(t) = \frac{\begin{array}{c}\text{\# of times tag } t \text{ appears}\\ \text{outside of a base NP}\end{array}}{\text{\# of occurrences of tag } t}$$

$$S_t(w) = \frac{\begin{array}{c}\text{\# of times tag } t \text{ and word } w\\ \text{appear outside of a base NP}\end{array}}{\begin{array}{c}\text{\# of times tag } t \text{ appears}\\ \text{outside of a base NP}\end{array}}$$

Parameter Selection. The values for λ_r, λ_p, λ_l, λ_t, and λ_w were selected automatically by comparing the performance of 44 selected combinations of parameter settings on the pruning corpus. The following combinations of bracket parameters (11) and nonbracket parameters (4) considered.

λ_r	0.8 0.1 0.1 0.6 0.5 0.5 0.3 0.4 0.5 0.4 0.33
λ_p	0.1 0.8 0.1 0.2 0.3 0.2 0.2 0.2 0.1 0.1 0.33
λ_l	0.1 0.1 0.8 0.2 0.2 0.3 0.5 0.4 0.4 0.5 0.33
λ_t	0.25 0.50 0.75 0.85
λ_w	0.75 0.50 0.25 0.15

Finding the Best Base NP Bracketing. We find the best path through the bracket graph using standard dynamic programming techniques. However, since different paths in the graph have different lengths (measured in number of arcs), each bracket arc score is first normalized for length: $\sqrt[|r|]{S(b)}$. This essentially makes a bracket that contains five words look like five single word arcs; thus it can be fairly compared to paths containing more but shorter arcs.

Compared to the quick linear time longest-match bracketer, the lexicalized bracketer is theoretically slower, since it compares multiple bracketings of the sentence. In practice, however, the number of alternative bracketings at any point is quite small and the lexicalized bracketer is only two or three times slower than the longest-match version.

Evaluation

We evaluated the lexicalized treebank grammars using the base NP corpora from our original study. The "Complex" corpus attempts to duplicate the base NPs used in the Ramshaw & Marcus study. The "Simple" corpus contains slightly less complicated base NPs — ones that may be better suited for use in some NLP applications, e.g., information extraction and information retrieval. In short, each Complex base NP corresponds to a non-recursive noun phrase in the Treebank parse. The Simple corpus further simplifies some of the Treebank base NPs by removing ambiguities that other components of the NLP system can be expected to handle: base NPs that contain conjunctions, prepositions, and leading or trailing adverbs and verbs are simplified.

The training, pruning, and testing sets are derived from the 25 sections of Wall Street Journal distributed with the Penn Treebank II. All experiments employ 5-fold cross validation. More specifically, the 25 sections are divided into five folds (sections 00–04, 05–09, and so on). In each of five runs, a different fold is used for testing the final, pruned rule set; three of the remaining

Corpus	Lexicalized Unpruned	Unlexicalized Unpruned	Pruned
Complex	88.7P/90.1R	18.3P/35.3R	88.7P/90.4R
Simple	91.5P/92.6R	22.1P/45.7R	92.1P/93.1R

Table 1: Performance of the Initial Lexicalized Grammar

	Lexicalized				Unlexicalized
Corpus	Unpruned	Threshold	Incremental	Baseline	Incremental
Complex	88.7P/90.1R	87.4P/90.5R	89.0P/90.9R	88.4P/90.5R	88.7P/90.4R
Simple	91.5P/92.6R	90.8P/93.2R	91.9P/93.6R	91.4P/93.1R	92.1P/93.1R

Table 2: The Effect of Pruning on the Lexicalized Grammar

folds comprise the training corpus (to create the initial rule set); and the fifth fold is the pruning corpus (which is also used for parameter selection). All results are averages across the five runs. Performance is measured in terms of precision and recall. Precision was described earlier — it is a standard measure of accuracy. Recall, on the other hand, is an attempt to measure coverage:

$$P = \frac{\text{\# of correct proposed NPs}}{\text{\# of proposed NPs}}$$

$$R = \frac{\text{\# of correct proposed NPs}}{\text{\# of NPs in the annotated text}}$$

We first evaluate the performance of the initial (i.e. unpruned) lexicalized grammars in Table 1. Not surprisingly, the lexicalized grammar (first column of results) is much more accurate than the initial unlexicalized grammar (column two) for both base NP data sets. When compared to the much simpler, unlexicalized grammars that were pruned using the incremental method (column three), however, we see slightly different results from the two corpora. For the Simple corpus, the lexicalized grammar performs somewhat below (-0.6P/-0.5R) the unlexicalized, but pruned, grammar of the original base NP study.[3] For the Complex corpus, the lexicalized grammar performs just comparably ($+0.0$P/-0.3R) to the unlexicalized pruned grammar. In general, these results suggest that lexicalization is not critical for recognizing very simple, relatively unambiguous constituents: For the simple base NP task, pruning appears to serve much the same function as lexicalization, in that both suppress the application of bad rules. Pruning, however, preserves the simplicity of the grammar and bracketing procedure. But as the complexity of the linguistic task increases, the importance of lexical information increases as well.

Given the strong performance of the initial lexicalized grammar, it is especially unclear whether pruning will have any effect for the lexicalized grammars. No existing statistical parsers for probabilistic context-free grammars, for example, include a grammar pruning step: it is assumed that the rule and lexical probabilities will effectively ignore "bad" rules. Table 2 shows our results for pruned lexicalized grammars. First, we see that the coarser threshold pruning (column two of results) performs worse than incremental pruning (column three) for both corpora. Not surprisingly, we originally showed that this result also holds for the unlexicalized grammars. When compared to the unpruned grammar (column one), threshold pruning lowers precision (-1.0) but raises recall ($+0.5$) for both corpora. Incremental pruning, on the other hand, increases both precision and recall (up to $+0.4$P/$+1.0$R) for both corpora, making the lexicalized grammar slightly better (for Complex) or just below (for Simple) the unlexicalized grammar (column five). Finally, incremental pruning yields better grammars than a baseline pruning strategy (column four) that discards all rules occurring only once during training as Charniak (1996) does.

That pruning can improve lexicalized grammars was unexpected: pruning considers each rule as an atomic source of errors — it does not prune the lexicalized versions of each rule independently. We believed that this might ultimately harm overall performance since lexicalized rules may vary in accuracy depending on the lexical content of the candidate bracket. Indeed, the more coarse-grained threshold pruning caused a drop in precision, while the finer-grained incremental pruning ultimately yielded an increase in performance. This suggests that a still more fine-grained pruning strategy — one that distinguishes the effects of the lexical content of candidate phrases — could be successfully employed with the lexicalized grammars.

The results in Table 2 were obtained using the parameter settings automatically selected on the pruning corpus. The final settings for all folds and for both corpora were $\lambda_r = 0.5, \lambda_p = 0.1, \lambda_l = 0.4, \lambda_t = 0.75, \lambda_w = 0.25$. Note that the value for λ_p — the lexical weight — is surprisingly low. In particular, λ_r indicates that the tag sequence rule scores are more important than the lexical information; while λ_l suggests that the "common-wisdom" heuristic of longest-match is actually quite relevant for this task. Accordingly, we evaluate the overall contribution of different sources of information in Ta-

[3]The results in the second and third columns of Table 1 differ slightly from those given in Cardie & Pierce (1998), due to changes in the training data.

Corpus	Rule Score + Length Score + Word Score	Rule Score + Length Score	Rule Score Only
Complex	88.7P/90.1R	88.2P/89.8R	84.7P/88.7R
Simple	91.5P/92.6R	91.6P/92.7R	88.5P/91.5R

Table 3: The Contribution of Sources of Information to a Lexicalized Base NP Grammar

Context	*Lexical Information*: No	*Lexical Information*: Yes
Yes	Memory-Based 91.6P/91.6R	Transformation-Based 93.1P/93.5R
No	Treebank-LM 88.7P/90.4R	Treebank-Lex 89.0P/90.9R

Table 4: Comparison of error-driven pruning of treebank grammars, transformation-based learning, and memory-based learning in terms of available knowledge sources for local context and lexical information

ble 3. The first column repeats the unpruned lexicalized grammar results. The second column shows results for the grammar without lexical information ($\lambda_p = \lambda_t = 0$). Then in the third column, the length term is also removed ($\lambda_l = 0$), leaving just the rule scores. The table shows that the longest-match heuristic contributes more to performance (more than 3.1P/1.1R) than lexical information (up to 0.5P/0.3R) for both corpora.

We conclude this section with examples from the Complex corpus that indicate the qualitative difference in performance between the unlexicalized (U) and lexicalized (L) versions of the base NP bracketer. The correct bracketing (C) is shown first for each example. In particular, note that the lexicalized bracketer correctly handles ambiguities that fool the original bracketer, such as some gerunds and conjunctions. And in cases where both bracketers err, the lexicalized one may produce more reasonable brackets.

C: *representing* [*general and administrative expenses*]
U: [*representing general and administrative expenses*]
L: *representing* [*general and administrative expenses*]

C: [*his previous real-estate investment and asset-management duties*]
U: [*his previous real-estate investment*] *and* [*asset-management duties*]
L: [*his previous real-estate investment and asset-management duties*]

C: [*president*] *and* [*chief operating officer*]
U: [*president and chief*] *operating* [*officer*]
L: [*president and chief operating officer*]

Comparison with Competing Approaches and Conclusions

To our knowledge, four different approaches to base noun phrase recognition have now been evaluated with respect to similar base NP corpora derived from the Penn Treebank — Ramshaw & Marcus's transformation-based bracketer, Argamon *et al.*'s MBSL, and the original and lexicalized versions of the treebank approach. As a result, we are now able to compare more or less directly a collection of trainable, corpus-based bracketing algorithms. In general, each method relies on slightly different information from the training corpus as depicted in Table 4. The Transformation-Based learner uses both lexical information and the surrounding context to make decisions about bracket boundary placement. MBSL's Memory-Based approach captures context in its tiles, but uses no lexical information. Our new lexicalized treebank grammar (Treebank-Lex) uses lexical information to score potential brackets, but no context. Finally, the simplest longest-match bracketer (Treebank-LM) uses neither lexical information nor context.

The results in Table 4 were not obtained on the same breakdown of training and test data; nevertheless, we can still attempt to glean from the table the role of context and lexicalization for base noun phrase identification. As we might expect, methods that employ lexical information perform better than those that do not; and methods that employ contextual information perform better than those that do not. The Transformation-Based bracketer, employing both, performs best in this group of algorithms. Comparing MBSL with Treebank-Lex, we might conclude that contextual information is more important than lexical information for base NP identification. However, MBSL has the added advantage of its ability to generalize its training data at recognition time.

In addition to performance differences, the four methods also vary in practical ways. For example, the Transformation-Based bracketer requires the most training time, with one pass through the training data for each candidate template for each new transformation. The Treebank methods need only a handful to a few hundred passes for pruning, while MBSL trains in a single pass over the training data. Regarding runtime speed, the Treebank-LM bracketer is the fastest, using a quick linear time algorithm. The Transformation-Based bracketer can be equally fast, provided that the transformation rules are precompiled into a single finite-state transducer (Roche & Schabes 1997). The Treebank-Lex bracketer is theoretically much slower than Treebank-LM due to its strategy of comparing multiple bracketings; but in practice it is only two to three times as slow. We believe MBSL to be the slowest bracketer, comparing multiple tiling covers to score each potential NP. In addition, the runtime space burden for MBSL is very large because the memory-base stores the entire training corpus. In contrast, the Transformation-Based and Treebank bracketers store only a few thousand transformation or grammar rules at runtime. Finally, in terms of portability, the two Treebank approaches are the only ones of the four that construct a grammar that may be modified for new genres of text without retraining.

In conclusion, we began with a simple and practical approach — treebank-derived tag-sequence grammars — for identifying simple syntactic phrases such as base noun phrases. We adapted this approach to incorporate lexical information into the selection of NP brackets, demonstrating its effectiveness for improving the accuracy of such grammars. We also showed that grammar pruning independently fulfills much the same function as lexicalization, reducing the impact of the noisiness of the training corpus, and in some cases proving to be more useful than lexical preferences. Somewhat surprisingly, we found that lexicalization and pruning can even be applied together with slightly increased performance over using them separately. Finally, we investigated the contributions of knowledge sources for the base noun phrase recognition task. For simple phrases, lexical information plays less of a role than expected; for complex phrases, lexicalization is more important; and the longest-match heuristic is quite useful for both simple and complex noun phrases. Although we focused only on base noun phrase recognition, we believe our results have implications for other probabilistic grammar-based approaches to natural language processing.

Acknowledgments. This work was supported in part by NSF Grants IRI–9624639 and GER–9454149. We thank Mitre for providing their part-of-speech tagger.

References

1996. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics.*

1998. *Proceedings of the 36th Annual Meeting of the ACL and COLING-98.*

Argamon, S.; Dagan, I.; and Krymolowski, Y. 1998. A Memory-Based Approach to Learning Shallow Natural Language Patterns. In *Proceedings of the 36th Annual Meeting of the ACL and COLING-98* (1998), 67–73.

Bikel, D.; Miller, S.; Schwartz, R.; and Weischedel, R. 1997. Nymble: A high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 194–201.

Bourigault, D. 1992. Surface grammatical analysis for the extraction of terminological noun phrases. In *Proceedings of the 15th International Conference on Computational Linguistics*, 977–981.

Brill, E. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* 21(4):543–565.

Cardie, C., and Pierce, D. 1998. Error-driven pruning of treebank grammars for base noun phrase identification. In *Proceedings of the 36th Annual Meeting of the ACL and COLING-98* (1998), 218–224. Available as cmp-lg/9808015.

Charniak, E. 1996. Treebank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1031–1036.

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

Church, K. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, 136–143. Association for Computational Linguistics.

Collins, M. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics* (1996), 184–191.

Justeson, J. S., and Katz, S. M. 1995. Technical terminology: Some linguistic properties and an algorithm for identification in text. *Natural Language Engineering* 1:9–27.

Marcus, M.; Marcinkiewicz, M.; and Santorini, B. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics* 19(2):313–330.

Ng, H., and Lee, H. 1996. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics* (1996), 40–47.

Ramshaw, L. A., and Marcus, M. P. 1998. Text chunking using transformation-based learning. In *Natural Language Processing Using Very Large Corpora*. Kluwer. Originally appeared in WVLC95, 82–94.

Roche, E., and Schabes, Y. 1997. Deterministic part-of-speech tagging with finite-state transducers. In Roche, E., and Schabes, Y., eds., *Finite-State Language Processing*. The MIT Press. chapter 7, 205–239.

Stanfill, C., and Waltz, D. 1986. Toward memory-based reasoning. *Communications of the ACM* 29:1213–1228.

Voutilainen, A. 1993. Nptool, a detector of english noun phrases. In *Proceedings of the Workshop on Very Large Corpora*, 48–57. Association for Computational Linguistics.