Parsing Noun Phrases in the Penn Treebank

David Vadas*
University of Sydney

James R. Curran**
University of Sydney

Noun phrases (NPs) are a crucial part of natural language, and can have a very complex structure. However, this NP structure is largely ignored by the statistical parsing field, as the most widely used corpus is not annotated with it. This lack of gold-standard data has restricted previous efforts to parse NPs, making it impossible to perform the supervised experiments that have achieved high performance in so many Natural Language Processing (NLP) tasks.

We comprehensively solve this problem by manually annotating NP structure for the entire Wall Street Journal section of the Penn Treebank. The inter-annotator agreement scores that we attain dispel the belief that the task is too difficult, and demonstrate that consistent NP annotation is possible. Our gold-standard NP data is now available for use in all parsers.

We experiment with this new data, applying the Collins (2003) parsing model, and find that its recovery of NP structure is significantly worse than its overall performance. The parser's F-score is up to 5.69% lower than a baseline that uses deterministic rules. Through much experimentation, we determine that this result is primarily caused by a lack of lexical information.

To solve this problem we construct a wide-coverage, large-scale NP Bracketing system. With our Penn Treebank data set, which is orders of magnitude larger than those used previously, we build a supervised model that achieves excellent results. Our model performs at 93.8% F-score on the simple NP task that most previous work has undertaken, and extends to bracket longer, more complex NPs that are rarely dealt with in the literature. We attain 89.14% F-score on this much more difficult task. Finally, we implement a post-processing module that brackets NPs identified by the Bikel (2004) parser. Our NP Bracketing model includes a wide variety of features that provide the lexical information that was missing during the parser experiments, and as a result, we outperform the parser's F-score by 9.04%.

These experiments demonstrate the utility of the corpus, and show that many NLP applications can now make use of NP structure.

1. Introduction

The parsing of noun phrases (NPs) involves the same difficulties as parsing in general. NPs contain structural ambiguities, just as other constituent types do, and resolving

* School of Information Technologies, University of Sydney, NSW 2006, Australia.
E-mail: dvadas1@it.usyd.edu.au.

** School of Information Technologies, University of Sydney, NSW 2006, Australia.
E-mail: james@it.usyd.edu.au.

Submission received: 23 April 2010; revised submission received: 17 February 2011; accepted for publication: 25 March 2011

these ambiguities is required for their proper interpretation. Despite this, statistical methods for parsing NPs have not achieved high performance until now.

Many Natural Language Processing (NLP) systems specifically require the information carried within NPs. Question Answering (QA) systems need to supply an NP as the answer to many types of factoid questions, often using a parser to identify candidate NPs to return to the user. If the parser cannot recover NP structure then the correct candidate may never be found, even if the correct dominating noun phrase has been found. As an example, consider the following extract:

> . . . as crude oil prices rose by 50%, a result of the. . .

and the question:

> The price of what commodity rose by 50%?

The answer *crude oil* is internal to the NP *crude oil prices*. Most commonly used parsers will not identify this internal NP, and will never be able to get the answer correct.

This problem also affects anaphora resolution and syntax-based machine translation. For example, Wang, Knight, and Marcu (2007) find that the flat tree structure of the Penn Treebank elongates the tail of rare tree fragments, diluting individual probabilities and reducing performance. They attempt to solve this problem by automatically binarizing the phrase structure trees. The additional NP annotation provides these SBSMT systems with more detailed structure, increasing performance. However, this SBSMT system, as well as others (Melamed, Satta, and Wellington 2004; Zhang et al. 2006), must still rely on a non-gold-standard binarization. Our experiments in Section 6.3 suggest that using supervised techniques trained on gold-standard NP data would be superior to these unsupervised methods.

This problem of parsing NP structure is difficult to solve, because of the absence of a large corpus of manually annotated, gold-standard NPs. The Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993) is the standard training and evaluation corpus for many syntactic analysis tasks, ranging from POS tagging and chunking, to full parsing. However, it does not annotate internal NP structure. The NP mentioned earlier, *crude oil prices*, is left flat in the Penn Treebank. Even worse, NPs with different structures (e.g., *world oil prices*) are given exactly the same annotation (see Figure 1). This means that any system trained on Penn Treebank data will be unable to model the syntactic and semantic structure inside base-NPs.

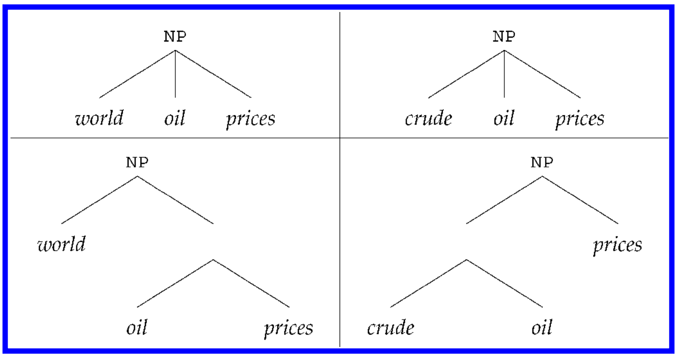


Figure 1
Parse trees for two NPs with different structures. The top row shows the identical Penn Treebank bracketings, and the bottom row includes the full internal structure.

Our first major contribution is a gold-standard labeled bracketing for every ambiguous noun phrase in the Penn Treebank. We describe the annotation guidelines and process, including the use of named entity data to improve annotation quality. We check the correctness of the corpus by measuring inter-annotator agreement and by comparing against DepBank (King et al. 2003). We also analyze our extended Treebank, quantifying how much structure we have added, and how it is distributed across NPs. This new resource will allow any system or corpus developed from the Penn Treebank to represent noun phrase structure more accurately.

Our next contribution is to conduct the first large-scale experiments on NP parsing. We use the newly augmented Treebank with the Bikel (2004) implementation of the Collins (2003) model. Through a number of experiments, we determine what effect various aspects of Collins's model, and the data itself, have on parsing performance. Finally, we perform a comprehensive error analysis which identifies the many difficulties in parsing NPs. This shows that the primary difficulty in bracketing NP structure is a lack of lexical information in the training data.

In order to increase the amount of information included in the NP parsing model, we turn to NP bracketing. This task has typically been approached with unsupervised methods, using statistics from unannotated corpora (Lauer 1995) or Web hit counts (Lapata and Keller 2004; Nakov and Hearst 2005). We incorporate these sources of data and use them to build large-scale supervised models trained on our Penn Treebank corpus of bracketed NPs. Using this data allows us to significantly outperform previous approaches on the NP bracketing task. By incorporating a wide range of features into the model, performance is increased by 6.6% F-score over our best unsupervised system.

Most of the NP bracketing literature has focused on NPs that are only three words long and contain only nouns. We remove these restrictions, reimplementing Barker's (1998) bracketing algorithm for longer noun phrases and combine it with the supervised model we built previously. Our system achieves 89.14% F-score on matched brackets. Finally, we apply these supervised models to the output of the Bikel (2004) parser. This post-processor achieves an F-score of 79.05% on the internal NP structure, compared to the parser output baseline of 70.95%.

This work contributes not only a new data set and results from numerous experiments, but also makes large-scale wide-coverage NP parsing a possibility for the first time. Whereas before it was difficult to even evaluate what NP information was being recovered, we have set a high benchmark for NP structure accuracy, and opened the field for even greater improvement in the future. As a result, downstream applications can now take advantage of the crucial information present in NPs.

2. Background

The internal structure of NPs can be interpreted in several ways, for example, the DP (determiner phrase) analysis argued by Abney (1987) and argued against by van Eynde (2006), treats the determiner as the head, rather than the noun. We will use a definition that is more informative for statistical modeling, where the noun—which is much more semantically indicative—acts as the head of the NP structure.

A noun phrase is a constituent that has a noun as its head,[1] and can also contain determiners, premodifiers, and postmodifiers. The head by itself is then an unsaturated

[1] The Penn Treebank also labels substantive adjectives such as *the rich* as NP, see Bies et al. (1995, §11.1.5)

NP, to which we can add modifiers and determiners to form a saturated NP. Or, in terms of X-bar theory, the head is an N-bar, as opposed to the fully formed NP. Modifiers do not raise the level of the N-bar, allowing them to be added indefinitely, whereas determiners do, making NPs such as **the the dog* ungrammatical.

The Penn Treebank annotates at the NP level, but leaves much of the N-bar level structure unspecified. As a result, most of the structure we annotate will be on unsaturated NPs. There are some exceptions to this, such as appositional structure, where we bracket the saturated NPs being apposed.

Quirk et al. (1985, §17.2) describe the components of a noun phrase as follows:

- The head is the central part of the NP, around which the other constituent parts cluster.
- The determinative, which includes predeterminers such as *all* and *both*; central determiners such as *the*, *a*, and *some*; and postdeterminers such as *many* and *few*.
- Premodifiers, which come between the determiners and the head. These are principally adjectives (or adjectival phrases) and nouns.
- Postmodifiers are those items after the head, such as prepositional phrases, as well as non-finite and relative clauses.

Most of the ambiguity that we deal with arises from premodifiers. Quirk et al. (1985, page 1243) specifically note that "premodification is to be interpreted . . . in terms of postmodification and its greater explicitness." Comparing *an oil man* to *a man who sells oil* demonstrates how a postmodifying clause and even the verb contained therein can be reduced to a much less explicit premodificational structure. Understanding the NP is much more difficult because of this reduction in specificity, although the NP can still be interpreted with the appropriate context.

2.1 Noun Phrases in the Penn Treebank

The Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993) annotates NPs differently from any other constituent type. This special treatment of NPs is summed up by the annotation guidelines (Bies et al. 1995, page 120):

> As usual, NP structure is different from the structure of other categories.

In particular, the Penn Treebank does not annotate the internal structure of noun phrases, instead leaving them flat. The Penn Treebank representation of two NPs with different structures is shown in the top row of Figure 1. Even though *world oil prices* is right-branching and *crude oil prices* is left-branching, they are both annotated in exactly the same way. The difference in their structures, shown in the bottom row of Figure 1, is not reflected in the underspecified Penn Treebank representation. This absence of annotated NP data means that any parser trained on the Penn Treebank is unable to recover NP structure.

Base-NP structure is also important for corpora derived from the Penn Treebank. For instance, CCGbank (Hockenmaier 2003) was created by semi-automatically converting the Treebank phrase structure to Combinatory Categorial Grammar (CCG) (Steedman

2000) derivations. Because CCG derivations are binary branching, they cannot directly represent the flat structure of the Penn Treebank base-NPs. Without the correct bracketing in the Treebank, strictly right-branching trees were created for all base-NPs. This is the most sensible approach that does not require manual annotation, but it is still incorrect in many cases. Looking at the following example NP, the CCGbank gold-standard is (a), whereas the correct bracketing would be (b).

```
(a)   (consumer ((electronics) and (appliances (retailing chain))))
(b)   ((((consumer electronics) and appliances) retailing) chain)
```

The Penn Treebank literature provides some explanation for the absence of NP structure. Marcus, Santorini, and Marcinkiewicz (1993) describe how a preliminary experiment was performed to determine what level of structure could be annotated at a satisfactory speed. This chosen scheme was based on the Lancaster UCREL project (Garside, Leech, and Sampson 1987). This was a fairly skeletal representation that could be annotated 100–200 words an hour faster than when applying a more detailed scheme. It did not include the annotation of NP structure, however.

Another potential explanation is that Fidditch (Hindle 1983, 1989)—the partial parser used to generate a candidate structure, which the annotators then corrected—did not generate NP structure. Marcus, Santorini, and Marcinkiewicz (1993, page 326) note that annotators were much faster at deleting structure than inserting it, and so if Fidditch did not generate NP structure, then the annotators were unlikely to add it.

The bracketing guidelines (Bies et al. 1995, §11.1.2) suggest a further reason why NP structure was not annotated, saying "it is often impossible to determine the scope of nominal modifiers." That is, Bies et al. (1995) claim that deciding whether an NP is left- or right-branching is difficult in many cases. Bies et al. give some examples such as:

```
(NP fake sales license)
(NP fake fur sale)
(NP white-water rafting license)
(NP State Secretary inauguration)
```

The scope of these modifiers is quite apparent. The reader can confirm this by making his or her own decisions about whether the NPs are left- or right-branching. Once this is done, compare the bracketing decisions to those made by our annotators, shown in this footnote.[2] Bies et al. give some examples that were more difficult for our annotators:

```
(NP week-end sales license)
(NP furniture sales license)
```

However this difficulty in large part comes from the lack of context that we are given. If the surrounding sentences were available, we expect that the correct bracketing would become more obvious. Unfortunately, this is hard to confirm, as we searched the corpus for these NPs, but it appears that they do not come from Penn Treebank text, and

[2] Right, left, left, left.

therefore the context is not available. And if the reader wishes to compare again, here are the decisions made by our annotators for these two NPs.[3]

Our position then, is that consistent annotation of NP structure is entirely feasible. As evidence for this, consider that even though the guidelines say the task is difficult, the examples they present can be bracketed quite easily. Furthermore, Quirk et al. (1985, page 1343) have this to say:

> Indeed, it is generally the case that obscurity in premodification exists only for the hearer or reader who is unfamiliar with the subject concerned and who is not therefore equipped to tolerate the radical reduction in explicitness that premodification entails.

Accordingly, an annotator with sufficient expertise at bracketing NPs should be capable of identifying the correct premodificational structure, except in domains they are unfamiliar with. This hypothesis will be tested in Section 4.1.

2.2 Penn Treebank Parsing

With the advent of the Penn Treebank, statistical parsing without extensive linguistic knowledge engineering became possible. The first model to exploit this large corpus of gold-standard parsed sentences was described in Magerman (1994, 1995). This model achieves 86.3% precision and 85.8% recall on matched brackets for sentences with fewer than 40 words on Section 23 of the Penn Treebank.

One of Magerman's important innovations was the use of deterministic head-finding rules to identify the head of each constituent. The head word was then used to represent the constituent in the features higher in the tree. This original table of head-finding rules has since been adapted and used in a number of parsers (e.g., Collins 2003; Charniak 2000), in the creation of derived corpora (e.g., CCGbank [Hockenmaier 2003]), and for numerous other purposes.

Collins (1996) followed up on Magerman's work by implementing a statistical model that calculates probabilities from relative frequency counts in the Penn Treebank. The conditional probability of the tree is split into two parts: the probability of individual base-NPs; and the probability of dependencies between constituents. Collins uses the CKY chart parsing algorithm (Kasami 1965; Younger 1967; Cocke and Schwartz 1970), a dynamic programming approach that builds parse trees bottom–up. The Collins (1996) model performs similarly to Magerman's, achieving 86.3% precision and 85.8% recall for sentences with fewer than 40 words, but is simpler and much faster.

Collins (1997) describes a cleaner, generative model. For a tree T and a sentence S, this model calculates the joint probability, $P(T, S)$, rather than the conditional, $P(T|S)$. This second of Collins's models uses a lexicalized Probabilistic Context Free Grammar (PCFG), and solves the data sparsity issues by making independence assumptions. We will describe Collins's parsing models in more detail in Section 2.2.1. The best performing model, including all of these extensions, achieves 88.6% precision and 88.1% recall on sentences with fewer than 40 words.

Charniak (1997) presents another probabilistic model that builds candidate trees using a chart, and then calculates the probability of chart items based on two values: the probability of the head, and that of the grammar rule being applied. Both of these

[3] Right, left.

are conditioned on the node's category, its parent category, and the parent category's head. This model achieves 87.4% precision and 87.5% recall on sentences with fewer than 40 words, a better result than Collins (1996), but inferior to Collins (1997). Charniak (2000) improves on this result, with the greatest performance gain coming from generating the lexical head's pre-terminal node before the head itself, as in Collins (1997).

Bikel (2004) performs a detailed study of the Collins (2003) parsing models, finding that lexical information is not the greatest source of discriminative power, as was previously thought, and that 14.7% of the model's parameters could be removed without decreasing accuracy.

Note that many of the problems discussed in this article are specific to the Penn Treebank and parsers that train on it. There are other parsers capable of recovering full NP structure (e.g., the PARC parser [Riezler et al. 2002]).

2.2.1 Collins's Models. In Section 5, we will experiment with the Bikel (2004) implementation of the Collins (2003) models. This will include altering the parser itself, and so we describe Collins's Model 1 here. This and the NP submodel are the parts relevant to our work.

All of the Collins (2003) models use a lexicalized grammar, that is, each nonterminal is associated with a head token and its POS tag. This information allows a better parsing decision to be made. However, in practice it also creates a sparse data problem. In order to get more reasonable estimates, Collins (2003) splits the generation probabilities into smaller steps, instead of calculating the probability of the entire rule. Each grammar production is framed as follows:

$$P(h) \rightarrow L_n(l_n)\dots L_1(l_1)H(h)R_1(r_1)\dots R_m(r_m) \tag{1}$$

where H is the head child, $L_n(l_n)\dots L_1(l_1)$ are its left modifiers, and $R_1(r_1)\dots R_m(r_m)$ are its right modifiers. Making independence assumptions between the modifiers and then using the chain rule yields the following expressions:

$$P_h(H|Parent, h) \tag{2}$$

$$\textstyle\prod_{i=1\dots n+1} P_l(L_i(l_i)|Parent, H, h) \tag{3}$$

$$\textstyle\prod_{i=1\dots m+1} P_r(R_i(r_i)|Parent, H, h) \tag{4}$$

The head is generated first, then the left and right modifiers, which are conditioned on the head but not on any other modifiers. A special STOP symbol is introduced (the $n+1^{th}$ and $m+1^{th}$ modifiers), which is generated when there are no more modifiers.

The probabilities generated this way are more effective than calculating over one very large rule. This is a key part of Collins's models, allowing lexical information to be included while still calculating useful probability estimates.

Collins (2003, §3.1.1, §3.2, and §3.3) also describes the addition of distance measures, subcategorization frames, and traces to the parsing model. However, these are not relevant to parsing NPs, which have their own submodel, described in the following section.

2.2.2 Generating NPs in Collins's Models. Collins's models generate NPs using a slightly different model to all other constituents. These differences will be important in Section 5, where we make alterations to the model and analyze its performance. For base-NPs, instead of conditioning on the head, the current modifier is dependent on the previous modifier, resulting in what is almost a bigram model. Formally, Equations (3) and (4) are changed as shown:

$$\prod_{i=1 \ldots n+1} P_l(L_i(l_i)|Parent, L_{i-1}(l_{i-1})) \quad (5)$$

$$\prod_{i=1 \ldots m+1} P_r(R_i(r_i)|Parent, R_{i-1}(r_{i-1})) \quad (6)$$

There are a few reasons given by Collins for this. Most relevant for this work is that because the Penn Treebank does not fully bracket NPs, the head is unreliable. When generating *crude* in the NP *crude oil prices*, we would want to condition on *oil*, the true head of the internal NP structure. However, *prices* is the head that would be found. Using the NP submodel thus results in the correct behavior. As Bikel (2004) notes, the model is not conditioning on the previous modifier *instead* of the head, the model is treating the previous modifier *as* the head. With the augmented Penn Treebank that we have created, the true head can now be identified. This may remove the need to condition on the previous modifier, and will be experimented with in Section 5.4.

The separate NP submodel also allows the parser to learn NP boundaries effectively, namely, that it is rare for words to precede a determiner in an NP. Collins (2003, page 602) gives the example *Yesterday the dog barked*, where conditioning on the head of the NP, *dog*, results in incorrectly generating *Yesterday* as part of the NP. On the other hand, if the model is conditioning on the previous modifier, *the*, then the correct STOP category is much more likely to be generated, as words do not often come before *the* in an NP.

Collins also notes that a separate X-bar level is helpful for the parser's performance. For this reason, and to implement the separate base-NP submodel, a preprocessing step is taken wherein NP brackets that do not dominate any other non-possessive NP nodes are relabeled as NPB. For consistency, an extra NP bracket is inserted around NPB nodes not already dominated by an NP. These NPB nodes are removed before evaluation. An example of this transformation can be seen here:

```
(S                                   (S
  (NP (DT The) (NN dog) )              (NP
  (VP (VBZ barks) ) )                    (NPB (DT The) (NN dog) ) )
                                       (VP (VBZ barks) ) )
```

2.3 NP Bracketing

Many approaches to identifying noun phrases have been explored as part of chunking (Ramshaw and Marcus 1995), but determining internal NP structure is rarely addressed. Recursive NP bracketing—as in the CoNLL 1999 shared task and as performed by Daumé III and Marcu (2004)—is closer, but still less difficult than full NP bracketing. Neither of these tasks require the recovery of full sub-NP structure, which is in part because gold-standard annotations for this task have not been available in the past.

Instead, we turn to the NP bracketing task as framed by Marcus (1980, page 253) and Lauer (1995), described as follows: given a three-word noun phrase like those here, decide whether it is left branching (a) or right branching (b):

(a) `((crude oil) prices)`

(b) `(world (oil prices))`

Most approaches to the problem use unsupervised methods, based on competing association strengths between pairs of words in the compound (Marcus 1980, page 253). There are two possible models to choose from: dependency or adjacency. The **dependency model** compares the association between words 1–2 to words 1–3, whereas the **adjacency model** compares words 1–2 to words 2–3. Both models are illustrated in Figure 2.

Lauer (1995) demonstrated superior performance of the dependency model using a test set of 244 (216 unique) noun compounds drawn from Grolier's encyclopedia. These data have been used to evaluate most research since. Lauer uses Roget's thesaurus to smooth words into semantic classes, and then calculates association between classes based on their counts in a body of text, also drawn from Grolier's. He achieves 80.7% accuracy using POS tags to identify bigrams in the training set.

Lapata and Keller (2004) derive estimates of association strength from Web counts, and only compare at a lexical level, achieving 78.7% accuracy. Nakov and Hearst (2005) also use Web counts, but incorporate additional counts from several variations on simple bigram queries, including queries for the pairs of words concatenated or joined by a hyphen. This results in an impressive 89.3% accuracy.

There have also been attempts to solve this task using supervised methods, even though the lack of gold-standard data makes this difficult. Girju et al. (2005) train a decision tree classifier, using 362 manually annotated NPs from the *Wall Street Journal* (WSJ) as training data, and testing on Lauer's data. For each of the three words in the NP, they extract five features from WordNet (Fellbaum 1998). This approach achieves 73.1% accuracy, although when they shuffled their WSJ data with Lauer's to create a new test and training split, performance increased to 83.1%. This may be a result of the ~10% duplication in Lauer's data set, however.

Barker (1998) describes an algorithm for bracketing longer NPs (described in Section 6.4) by reducing the problem to making a number of decisions on three word NPs. This algorithm is used as part of an annotation tool, where three-word NPs for which no data are available are presented to the user. Barker reports accuracy on these three-word NPs only (because there is no gold-standard for the complete NPs), attaining 62% and 65% on two different data sets.

In this section, we have described why the Penn Treebank does not internally annotate NPs, as well as how a widely used parser generates NP structure. The following section will detail how we annotated a corpus of NPs, creating data for both a PCFG parser and an NP bracketing system.

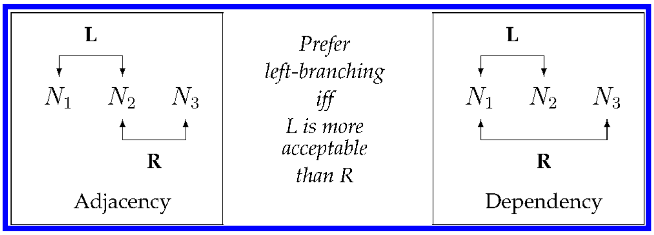


Figure 2
The associations compared by the adjacency and dependency models, from Lauer (1995).

3. Annotation Process

The first step to statistical parsing of NPs is to create a gold-standard data set. This section will describe the process of manually annotating such a corpus of NP structure. The data will then be used in the parsing experiments of Section 5 and the NP Bracketing experiments in Section 6. Extending the Penn Treebank annotation scheme and corpus is one of the major contributions of this article.

There are a handful of corpora annotated with NP structure already, although these do not meet our requirements. DepBank (King et al. 2003) fully annotates NPs, as does the Briscoe and Carroll (2006) reannotation of DepBank. This corpus consists of only 700 sentences, however. The Redwoods Treebank (Oepen et al. 2002) also includes NP structure, but is again comparatively small and not widely used in the parsing community. The Biomedical Information Extraction Project (Kulick et al. 2004) introduces the use of NML nodes to mark internal NP structure in its Addendum to the Penn Treebank Bracketing Guidelines (Warner et al. 2004). This corpus is specifically focused on biomedical text, however, rather than newspaper text. We still base our approach to bracketing NP structure on these biomedical guidelines, as the grammatical structure being annotated remains similar.

We chose to augment the WSJ section of the Penn Treebank with the necessary NP structure, as it is the corpus most widely used in the parsing field for English. This also meant that the NP information would not need to be imported from a separate model, but could be included into existing parsers and their statistical models with a minimum of effort. One principle we applied during the augmentation process was to avoid altering the original Penn Treebank brackets. This meant that results achieved with the extended corpus would be comparable to those achieved on the original, excluding the new NP annotations.

The manual annotation was performed by the first author, and a computational linguistics PhD student also annotated Section 23. This helped to ensure the reliability of the annotations, by allowing inter-annotator agreement to be measured (see Section 4.1). This also maximized the quality of the section used for parser testing. Over 60% of sentences in the corpus were manually examined during the annotation process.

3.1 Annotation Guidelines

We created a set of guidelines in order to aid in the annotation process and to keep the result consistent and replicable. These are presented in full in Appendix A, but we will also present a general description of the guidelines here, together with a number of examples.

Our approach is to leave right-branching structures unaltered, while labeled brackets are inserted around left-branching structures.

```
(NP (NN world) (NN oil) (NNS prices) )

(NP (NML (NN crude) (NN oil) )
  (NNS prices) )
```

Left- and right-branching NPs are now differentiated. Although explicit brackets are not added to right-branching NPs, they should now be interpreted as having the following implicit structure:

```
(NP (NN world)
  (NODE (NN oil) (NNS prices) ) )
```

This representation was used in the biomedical guidelines, and has many advantages. By keeping right-branching structure implicit, the tree does not need to be binarized. Binarization can have a harmful effect on parsers using PCFGs, as it reduces the context-sensitivity of the grammar (Collins 2003, page 621). It also reduces the amount of clutter in the trees, making them easier to view and annotate. Right-branching structure can still be added automatically if required, as we experiment with in Section 5.5. Not inserting it, however, makes the annotator's task simpler.

The label of the newly created constituent is NML (**nominal modifier**), as in the example above, or JJP (**adjectival phrase**), depending on whether its head is a noun or an adjective. Examples using the JJP label are shown here:

```
(NP (JJP (JJ dark) (JJ red) )
  (NN car) )

(NP (DT the)
  (JJP (JJS fastest) (VBG developing) )
  (NNS trends) )
```

Rather than this separate JJP label, the biomedical treebank replicates the use of the ADJP label in the original Penn Treebank. We wanted to be able to distinguish the new annotation from the old in later experiments, which required the creation of this additional label. JJPs can easily be reverted back to ADJP, as we will experiment with in Section 5.2.

Non-base-NPs may also need to be bracketed, as shown:

```
(NP-SBJ
  (NML (JJ former)
    (NAC (NNP Ambassador)
      (PP (TO to)
        (NP (NNP Costa) (NNP Rica) ) ) ) )
  (NNP Francis) (NNP J.) (NNP McNeil) )
```

In this example, we join *former* and the NAC node, as he is formerly the *Ambassador*, not formerly *Mr. McNeil*.

Many coordinations need to be bracketed, as in the following examples:

```
(NP (DT the)
  (NML (NNPS Securities)
    (CC and) (NNP Exchange) )
  (NNP Commission) )

(NP (PRP$ its)
  (JJP (JJ current)
    (CC and) (JJ former) )
  (NNS ratepayers) )
```

Without these brackets, the NP's implicit structure, as shown here, would be incorrect.

```
(NP (DT the)
  (NODE
    (NODE (NNPS Securities) )
    (CC and)
    (NODE (NNP Exchange) (NNP Commission) ) ) )
```

The erroneous meaning here is *the Securities* and *the Exchange Commission*, rather than the correct *the Securities Commission* and *the Exchange Commission*. There is more detail on how coordinations are bracketed in Appendix A.2.1

As can be seen from these examples, most of the our annotation is concerned with how premodifiers attach to each other and to their head.

3.1.1 Difficult Cases. During the annotation process, we encountered a number of NPs that were difficult to bracket. The main cause of this difficulty was technical jargon, for example, in the phrase *senior subordinate reset discount debentures*. The Penn Treebank guidelines devote an entire section to this *Financialspeak* (Bies et al. 1995, §23). The biomedical guidelines similarly contain some examples that are difficult for a non-biologist to annotate:

```
liver cell mutations
p53 gene alterations
ras oncogene expression
polymerase chain reaction
```

Even these NPs were simple to bracket for an expert in the biological domain, however. We did find that there were relatively few NPs that the annotators clearly understood, but still had difficulty bracketing. This agrees with our hypothesis in Section 2.1, that modifier scope in NPs is resolvable.

For those difficult-to-bracket NPs that were encountered, we bracket what structure is clear and leave the remainder flat. This results in a right-branching default. The biomedical guidelines (Warner et al. 2004, §1.1.5) also take this approach, which can be compared to how ambiguous attachment decisions are bracketed in the Penn Treebank and in the Redwoods Treebank (Oepen et al. 2002). Bies et al. (1995, §5.2.1) says "the default is to attach the constituent at the highest of the levels where it can be interpreted."

3.2 Annotation Tool

We developed a bracketing tool to identify ambiguous NPs and present them to an annotator for disambiguation. An ambiguous NP is any (possibly non-base) NP with three or more contiguous children that are either single words or another NP. Certain common patterns, such as three words beginning with a determiner, were observed as being entirely unambiguous during the initial phase of the annotation. Because of this, they are filtered out by the tool. The complete list of patterns is: `* CC *`, `$ * * -NONE-`, `DT * *`, `PRP$ * *`, and `* * POS`. The latter pattern also inserts a `NML` bracket around the first two tokens.

In order to better inform the annotator, the tool also displayed the entire sentence surrounding the ambiguous NP. During the annotation process, most NPs could be bracketed without specifically reading this information, because the NP structure was clear and/or because the annotator already had some idea of the article's content from the NPs (and surrounding context) shown previously. In those cases where the surrounding sentence provided insufficient context for disambiguation, it was typically true that no amount of surrounding context was informative. For these NPs, the principle of leaving difficult cases flat was applied. We did not mark flat NPs during the annotation process (it is a difficult distinction to make) and so cannot provide a figure for how many there are.

3.2.1 Automatic Bracketing Suggestions. We designed the bracketing tool to automatically suggest a bracketing, using rules based mostly on named entity tags. These NER tags are drawn from the BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein 2005). This corpus of gold-standard data annotates 29 different entity tags. Some of the NER tags have subcategories, for example, **GPE** (Geo-Political Entity) is divided into **Country**, **City**, **State/Province** and **Other**, however we only use the coarse tags for the annotation tool suggestions.

This NER information is useful, for example, in bracketing the NP *Air Force contract*. Because *Air Force* is marked as an organization, the tool can correctly suggest that the NP is left-branching. Using NER tags is more informative than simply looking for NNP POS tags, as there are many common nouns that are entities; for example, *vice president* is a **PER_DESC** (person descriptor).

The tool also suggests bracketings based on the annotator's previous decisions. Whenever the annotator inserts a bracket, the current NP and its structure, together with the label and placement of the new bracket, is stored. Then, whenever the same NP and structure is seen in the future, the same bracketing is suggested. This source of suggestions is particularly important, as it helps to keep the annotator consistent.

Other suggestions are based on gazetteers of common company and person name endings. Preliminary lists were generated automatically by searching for the most frequently occurring final tokens in the relevant named entities. Some of the most common examples are *Co.* and *Inc* for companies and *Jr* and *III* for people's names. There were also some incorrect items that were removed from the lists by manual inspection.

The guidelines also mandate the insertion of nodes around brackets and speech marks (see Appendix A.2.2 and A.2.3). These are detected automatically and included in the suggestion system accordingly. Unbalanced quotes do not result in any suggestions.

The last source of suggestions is final possessives, as in *John Smith's*. In these cases, a bracket around the possessor *John Smith* is suggested.

It should be noted that using this suggestion feature of the annotation tool may bias an annotator towards accepting an incorrect decision. The use of previous decisions in particular makes it much easier to always choose the same decision. We believe it is worth the trade-off of using the suggestions, however, as it allows faster, more consistent annotation.

3.3 Annotation Post-Processes

In order to increase the reliability of the corpus, a number of post-processes have been carried out since the annotation was first completed. Firstly, 915 NPs were marked by the annotator as difficult during the main annotation phase. In discussion with two other experts, the best bracketing for these NPs was determined. Secondly, the annotator identified 241 phrases that occurred numerous times and were non-trivial to bracket. These phrases were usually idiomatic expressions like *U.S. News & World Report* and/or featured technical jargon as in *London Interbank Offered Rate*. An extra pass was made through the corpus, ensuring that every instance of these phrases was bracketed consistently.

The main annotator made another pass (from version 0.9 to 1.0) over the corpus in order to change the standard bracketing for coordinations, speech marks, and brackets. These changes were aimed at increasing consistency and bringing our annotations more in line with the biomedical guidelines (Kulick et al. 2004). For example, *royalty and rock stars* is now bracketed the same way as *rock stars and royalty*. For more detail, see Sections A.2.1, A.2.2, and A.2.3 in the annotation guidelines appendix.

Only those NPs that had at least one bracket inserted during the first pass were manually inspected during this pass. NPs with a conjunction followed by multiple tokens, such as *president and chief executive officer*, also needed to be reannotated. By only reanalyzing this subset of ambiguous NPs, the annotator's workload was reduced, while still allowing for a number of simple errors to be noted and corrected.

Lastly, we identified all NPs with the same word sequence and checked that they were always bracketed identically. Those that differed from the majority bracketing were manually reinspected and corrected as necessary. However, even after this process, there were still 48 word sequences by type (201 by token) that were inconsistent. In these remaining cases, such as the NP below:

```
(NP-TMP (NML (NNP Nov.) (CD 15))     (NP-TMP (NP (NNP Nov.) (CD 15))
   (, ,)                                (, ,)
   (CD 1999))                           (CD 1999))
```

we were inconsistent in inserting the NML node (shown on the left) because the Penn Treebank sometimes already has the structure annotated under an NP node (shown on the right). Since we do not make changes to existing brackets, we cannot fix these cases. This problem may be important later on, as a statistical parser will have difficulty learning whether it is appropriate to use an NML or NP label.

3.4 Annotation Time

Annotation initially took over 9 hours per section of the Treebank. With practice, however, this was reduced to about 3 hours per section. Each section contains around 2,500 ambiguous NPs (i.e., annotating took approximately 5 seconds per NP). Most NPs require no bracketing, or fit into a standard pattern which the annotator soon becomes accustomed to, hence the task can be performed quite quickly.

As a comparison, during the original creation of the Treebank, annotators performed at 375–475 words per hour after a few weeks, and increased to about 1,000 words per hour after gaining more experience (Marcus, Santorini, and Marcinkiewicz 1993). For our annotations, we would expect to be in the middle of this range, as the task was not large enough to get more than a month's experience, or perhaps faster as there is less structure to annotate. The actual figure, calculated by counting each word in every NP shown, is around 800 words per hour. This matches the expectation quite well.

4. Corpus Analysis

Looking at the entire Penn Treebank corpus, the annotation tool finds 60,959 ambiguous NPs out of the 432,639 NPs in the corpus (14.09%). Of these, 23,129 (37.94%) had brackets inserted by the annotator. This is as we expect, as the majority of NPs are right-branching. Of the brackets added, 26,372 were NML nodes, and 894 were JJP.

To compare, we can count the number of existing NP and ADJP nodes found in the NPs that the bracketing tool presents. We find there are 32,772 NP children, and 579 ADJP, which is quite similar to the number and proportion of nodes we have added. Hence, our annotation process has introduced almost as much structural information into NPs as there was in the original Penn Treebank.

Table 1 shows the most common POS tag sequences for NP, NML, and JJP nodes, over the entire corpus. An example is given showing typical words that match the POS tags. For NML and JJP, the example shows the complete NP node, rather than just the NML or JJP bracket. It is interesting to note that RB JJ sequences are annotation errors in the original Treebank, and should have an ADJP bracket already.

Table 1
The most common POS tag sequences in the NP annotated corpus. The examples show a complete NP, and thus the POS tags for NML and JJP match only the bracketed words.

LABEL	COUNT	POS TAGS	EXAMPLE
NP	3,557	NNP NNP NNP	*John A. Smith*
	2,453	DT NN POS	*(the dog) 's*
	1,693	JJ NN NNS	*high interest rates*
NML	8,605	NNP NNP	*(John Smith) Jr.*
	2,475	DT NN	*(the dog) 's*
	1,652	NNP NNP NNP	*(A. B. C.) Corp*
JJP	162	‘ ‘ JJ ’ ’	*(" smart ") cars*
	120	JJ CC JJ	*(big and red) apples*
	112	RB JJ	*(very high) rates*

4.1 Inter-Annotator Agreement

To determine the correctness and consistency of our corpus, we calculate inter-annotator agreement on Section 23. Note that the second annotator was following version 0.9 of the bracketing guidelines, and since then the guidelines have been updated to version 1.0. Because of this, we can only analyze the 0.9 version of the corpus, that is, before the primary annotator made the second pass mentioned in Section 3.3.[4] This is not problematic, as the definition of what constitutes an NML or JJP node has not changed, only their representation in the corpus. That is, the dependencies that can be drawn from the NPs remain the same.

We have not calculated a kappa statistic, a commonly used measure of inter-annotator agreement, as it is difficult to apply to this task. This is because the bracketing of an NP cannot be divided into two choices; there are far more possibilities for NPs longer than three words. Whether the evaluation is over brackets or dependencies, there is always structure that the annotator has made an implicit decision not to add, and counting these true negatives is a difficult task. The true negative count cannot be taken as zero either, as doing so makes the ratios and thus the final kappa value uninformative.

Instead, we measure the proportion of matching brackets and (unlabeled) dependencies between annotators, taking one as a gold standard and then calculating precision, recall, and F-score. For the brackets evaluation, we count only the newly added NML and JJP brackets, not the enclosing NP or any other brackets. This is because we want to evaluate our annotation process and the structure we have added, not the pre-existing Penn Treebank annotations. The dependencies are generated by assuming the head of a constituent is the right-most token, and then joining each modifier to its head. This is equivalent to adding explicit right-branching brackets to create a binary tree. The number of dependencies is fixed by the length of the NP, so the dependency precision and recall are the same.

Table 2 shows the results, including figures from only those NPs that have three consecutive nouns. Noun compounds such as these have a high-level of ambiguity (as

[4] Although the subsequent consistency checks described there *had* been carried out, and were applied again afterwards.

Table 2
Agreement between annotators, before and after discussion and revision. Two evaluations are shown: matched brackets of the newly added NML and JJP nodes, and automatically generated dependencies for all words in the NP.

	PREC.	RECALL	F-SCORE
Brackets	89.17	87.50	88.33
Dependencies	96.40	96.40	96.40
Brackets, NPs with three consecutive nouns only	87.46	91.46	89.42
Dependencies, NPs with three consecutive nouns only	92.59	92.59	92.59
Brackets, revised	97.56	98.03	97.79
Dependencies, revised	99.27	99.27	99.27

will be shown later in Table 14), so it is interesting to compare results on this subset to those on the corpus in full. Table 2 also shows the result after cases of disagreement were discussed and the annotations revised.

In all cases, matched brackets give a lower inter-annotator agreement F-score. This is because it is a harsher evaluation, as there are many NPs that both annotators agree should have no additional bracketing that are not taken into account by the metric. For example, consider an NP that both annotators agree is right-branching:

```
(NP (NN world) (NN oil) (NNS prices))
```

The F-score is not increased by the matched bracket evaluation here, as there is no NML or JJP bracket and thus nothing to evaluate. A dependency score, on the other hand, would find two matching dependencies (between *world* and *prices* and *oil* and *prices*), increasing the inter-annotator agreement measure accordingly.

We can also look at exact matching on NPs, where the annotators originally agreed in 2,667 of 2,908 cases (91.71%), or 613 of 721 (85.02%) NPs that had three consecutive nouns. After revision, the annotators agreed in 2,864 of 2,908 cases (98.49%). Again, this is a harsher evaluation as partial agreement is not taken into account.

All of these inter-annotator figures are at a high level, thus demonstrating that the task of identifying nominal modifier scope can be performed consistently by multiple annotators. We have attained high agreement rates with all three measures, and found that even difficult cases could be resolved by a relatively short discussion.

The bracketing guidelines were revised as a result of the post-annotation discussion, to clarify those cases where the disagreements had occurred. The disagreements after revision occurred for a small number of repeated instances, such as:

```
(NP (NNP Goldman)                    (NP
  (, ,)                                (NML (NNP Goldman)
  (NNP Sachs)                            (, ,)
  (CC &) (NNP Co) )                      (NNP Sachs) )
                                       (CC &) (NNP Co) )
```

The second annotator felt that *Goldman , Sachs* should form its own NML constituent, whereas the first annotator did not.

We would like to be able to compare our inter-annotator agreement to that achieved in the original Penn Treebank project. Marcus, Santorini, and Marcinkiewicz (1993) describe a 3% estimated error rate for their POS tag annotations, but unfortunately, no figure is given for bracketing error rates. As such, a meaningful comparison between the NP annotations described here and the original Penn Treebank cannot be made.

We can compare against the inter-annotator agreement scores in Lauer (1995, §5.1.7). Lauer calculates a pair-wise accuracy between each of seven annotators, and then averages the six numbers for each annotator. This results in agreement scores between 77.6% and 82.2%. These figures are lower than those we have reported here, although Lauer only presented the three words in the noun compound with no context. This makes the task significantly harder, as can be seen from the fact the annotators only achieve between 78.7% and 86.1% accuracy against the gold standard. Considering this, it is not surprising that the annotators were not able to come to the same level of agreement that the two annotators in our process reached.

4.2 DepBank Agreement

Another approach to measuring annotator reliability is to compare with an independently annotated corpus of the same text. We use the Briscoe and Carroll (2006) version of the PARC700 Dependency Bank (King et al. 2003). These 560 sentences from Section 23 are annotated with labeled dependencies, and are used to evaluate the RASP parser.

Some translation is required to compare our brackets to DepBank dependencies, as this is not a trivial task. We map the brackets to dependencies by finding the head of the NP, using the Collins (1999) head-finding rules, and then creating a dependency between each other child's head and this head. The results are shown in Table 3. We give two evaluation scores, the dependencies themselves and how many NPs had all their dependencies correct. The second evaluation is tougher, and so once again the dependency numbers are higher than those at the NP level. And although we cannot evaluate matched brackets as we did for inter-annotator agreement, we can (in the bottom two rows of the table) look only at cases where we have inserted some annotations, which is similar in effect. As expected, these are more difficult cases and the score is not as high.

The results of this analysis are better than they appear, as performing a cross-formalism conversion to DepBank does not work perfectly. Clark and Curran (2007) found that their conversion method to DepBank only achieved 84.76% F-score on labeled dependencies, even when using gold-standard data. In the same way, our agreement figures could not possibly reach 100%. Accordingly, we investigated the errors manually to determine their cause, with the most common results shown in Table 4.

True disagreement between the Briscoe and Carroll (2006) annotations and ours is only the second most common cause. In the example in Table 4, the complete sentence

Table 3
Agreement with DepBank. Two evaluations are shown: over-all dependencies, and where all dependencies in an NP must be correct. The bottom two rows exclude NPs where no NML or JJP annotation was added.

	MATCHED	TOTAL	%
By dependency	1,027	1,114	92.19
By noun phrase	358	433	82.68
By dependency, only annotated NPs	476	541	87.99
By noun phrase, only annotated NPs	150	203	73.89

Table 4
Disagreement analysis with DepBank, showing how many dependencies were not matched.

ERROR TYPE	COUNT	EXAMPLE NP
Company name post-modifier	26	*Twenty-First Securities Corp*
True disagreement	25	*mostly real estate*
Head finding error	21	*Skippy the Kangaroo*

is: *These "clean-bank" transactions leave the bulk of bad assets, mostly real estate, with the government, to be sold later.* We annotated *mostly real estate* as a right-branching NP, that is, with dependencies between *mostly* and *estate* and *real* and *estate*. Briscoe and Carroll form a dependency between *mostly* and *real*.

The largest source of disagreements arises from how company names are bracketed. Whereas we have always separated the company name from post-modifiers such as *Corp* and *Inc*, DepBank does not in most cases. The other substantial cause of annotation discrepancies is a result of the head-finding rules. In these cases, the DepBank dependency will often be in the opposite direction of the Penn Treebank one, or the head found by Collins's rules will be incorrect. For example, in the NP *Skippy the Kangaroo*, the Collins's head-finding rules identify *Kangaroo* as the head, whereas the DepBank head is *Skippy*. In both cases, a dependency between the two words is created, although the direction is different and so no match is found.

Even without taking these problems into account, these results show that consistently and correctly bracketing noun phrase structure is possible, and that inter-annotator agreement is at an excellent level.

4.3 Evaluating the Annotation Tool's Suggestions

This last analysis of our corpus evaluates the annotation tool's suggestion feature. This will serve as a baseline for NP bracketing performance in Section 5, and will be a much stronger baseline than making all NPs left- or right-branching. A left-branching baseline would perform poorly, as only 37.94% of NPs had left-branching structure. A right-branching baseline would be even worse as no brackets would be inserted, resulting in an F-score of 0.0%.

The annotation tool was run over the entire Penn Treebank in its original state. Suggestions were automatically followed and no manual changes were made. All the suggestion rules (described in Section 3.2.1) were used, except for those from the annotator's previous bracketings, as these would not be available unless the annotation had already been completed. Also note that these experiments use gold-standard NER data; we expect that automatically generated NER tags would not perform as well. The results in Table 5 show that in all cases, the suggestion rules have high precision and low recall. NER-based features, for example, are only helpful in NPs that dominate named entities, although whenever they can be applied, they are almost always correct.

The subtractive analysis shows that each of the suggestion types increases performance, with NER and company and name endings providing the biggest gains. Surprisingly, precision improves with the removal of the NER suggestion type. We suspect that this is caused by some of the annotation choices in the BBN corpus that do not align well with the parse structure. For example, in *Mr Vinken*, the words are annotated as **O** and **PERSON** respectively, rather than having **PERSON** on both words. Conversely, all three tokens in *a few years* are annotated as **DATE**, even though *years* is the only date-related word.

Note that all of the results in Table 5, except for the last two lines, are evaluating over the entire corpus, as there was no need for training data. With this baseline, we have set a significant challenge for finding further improvement.

5. Statistical Parsing

In the previous section, we described the augmentation of the Penn Treebank with NP structure. We will now use this extended corpus to conduct parsing experiments. We use the Bikel (2004) implementation of the Collins (2003) models, as it is a widely used and well-known parser with state-of-the-art performance. It is important to make the distinction between Collins's and Bikel's parsers, as they are not identical. The same is true for their underlying models, which again have slight differences. We use Bikel's *parser* in all of our experiments, but will still refer to Collins's *models* for the most part.

We compare the parser's performance on the original Penn Treebank and the new NML and JJP bracketed version. We report the standard Parseval measures (Black et al. 1991) labeled bracket precision, recall, and F-scores over all sentences. Sections 02–21 are used for training, Section 00 for development, and testing is carried out on Section 23.

5.1 Initial Experiments

Table 6 shows the results of Section 00. The first row comes from training and evaluating on the original Penn Treebank, and the next three are all using the extended NP corpus. The first of these, **Original structure**, evaluates only the brackets that existed before the NP augmentation. That is, the NML and JJP brackets are removed before calculating these figures, in the same way that the NPB brackets added as part of Collins's parsing process are excised. The next figures, for **NML and JJP brackets only**, work in the opposite manner, with all brackets besides NML and JJP being ignored. The final row shows the results when all of the brackets—NMLs, JJPs, and the original structure—are evaluated.

These figures supply a more detailed picture of how performance has changed, showing that although the new brackets make parsing marginally more difficult overall (by about 0.5% in F-score), accuracy on the original structure is only negligibly worse.

Table 5
Suggestion rule performance. The middle group shows a subtractive analysis, removing individual suggestion groups from the All row. The final two rows are on specific sections; all other figures are calculated over the entire corpus.

SUGGESTIONS USED	PREC.	RECALL	F-SCORE
NER only	94.16	32.57	48.40
All	94.84	54.86	69.51
−NER	97.46	41.31	58.02
−Company and name endings	94.55	41.42	57.60
−Brackets and speech marks	95.03	50.62	66.05
−Possessives	94.51	50.95	66.20
All, Section 00	95.64	59.36	73.25
All, Section 23	94.29	56.81	70.90

Table 6
Performance achieved with the Bikel (2004) parser, initial results on development set.

	PREC.	RECALL	F-SCORE
Original PTB	88.88	88.85	88.86
Original structure	88.81	88.88	88.85
NML and JJP brackets only	76.32	60.42	67.44
All brackets	88.55	88.15	88.35

The new NML and JJP brackets are the cause of the performance drop, with an F-score more than 20% lower than the overall figure. This demonstrates the difficulty of parsing NPs.

The all-brackets result actually compares well to the original Penn Treebank model, as the latter is not recovering or being evaluated on NP structure and as such, has a much easier task. However the parser's performance on NML and JJP brackets is surprisingly poor. Indeed, the figure of 67.44% is more than 5% lower than the baseline established using the annotation tool's suggestions (see Table 5). The suggestions were in part based on NER information that the parser does not possess, but we would still expect the parser to outperform a set of deterministic rules. The rest of this section will describe a number of attempts to improve the parser's performance by altering the data being used and the parser model itself.

5.2 Relabeling NML and JJP

Bikel's parser does not come inbuilt with an expectation of NML or JJP nodes in the treebank, and these new labels could cause problems. For example, head-finding for these constituents is undefined. Further, changing the structure of NPs (which are already treated differently in many aspects of Collins's model) also has deeper implications, as we shall see. In an attempt to remove any complications introduced by the new labels, we ran an experiment where the new NML and JJP labels were relabeled as NP and ADJP. These are the labels that would be given if NPs were originally bracketed with the rest of the Penn Treebank. This relabeling means that the model does not have to discriminate between two different types of noun and adjective structure, and for this reason we might expect to see an increase in performance. This approach is also easy to implement, and negates the need for any change to the parser itself.

The figures in Table 7 show that this is not the case, as the all-brackets F-score has dropped by almost half a percent, compared to the numbers in Table 6. To evaluate the NML and JJP brackets only, we compare against the corpus without relabeling, and whenever a test NP matches a gold NML we count it as a correct bracketing. The same is done for ADJP and JJP brackets. However, only recall can be measured in this way, and not precision, as the parser does not produce NML or JJP brackets that can be evaluated.

These nodes can only be known when they have already been matched against the gold standard, which falsely suggests a precision of 100%. The incorrect NML and JJP nodes are hidden by incorrect NP or ADJP nodes and the difference cannot be recovered. Thus the NML and JJP brackets difference in Table 7 is for recall, not F-score. This also means that the figures given for the original structure are not entirely accurate, as the original NPs cannot be distinguished from the NMLs we annotated and have converted to NPs. This explains why precision drops by 0.89%, whereas recall is only 0.20% lower.

Table 7
Performance achieved with the Bikel (2004) parser and relabeled brackets. The DIFF column compares against the initial results in Table 6.

	PREC	RECALL	F-SCORE	DIFF
Original structure	87.92	88.68	88.30	−0.55
NML and JJP brackets only	–	53.54	–	−6.88
All brackets	88.09	87.77	87.93	−0.42

Despite all these complications, the decreases in performance on every evaluation make it clear that the relabeling has not been successful. We carried out a visual inspection of the errors that were made in this experiment, which hadn't been made when the NP and NML labels were distinct. It was noticeable that many of these errors occurred when a company name or other entity needed to be bracketed, such as W.R. Grace in the following gold-standard NP:

```
(NP
  (ADVP (RB formerly) )
  (DT a) (NML (NNP W.R.) (NNP Grace) )
  (NN vice) (NN chairman) )
```

The parser output had no bracket around *W.R. Grace*.

We conclude that the model was not able to generalize a rule that multiple tokens with the NNP POS tag should be bracketed. Even though NML brackets often follow this rule, NPs do not. As a result, the distinction between the labels should be retained, and we must change the parser itself to deal with the new labels properly.

5.3 Head-Finding Rules

The first and simplest change we made was to create head-finding rules for NML and JJP constituents. In the previous experiments, these nodes would be covered by the catch-all rule, which simply chooses the left-most child as the head. This is incorrect in most NMLs, where the head is usually the right-most child. To define the NML and JJP rules, we copy those for NPs and ADJPs, respectively. We also add to the rules for NPs, so that child NML and JJP nodes can be recursively examined, in the same way that NPs and ADJPs are. This change is not needed for other labels, as NMLs and JJPs only exist under NPs. We ran the parser again with this change, and achieved the results in Table 8. The differences shown are against the original results from Table 6.

Table 8
Performance achieved with the Bikel (2004) parser and correct head-finding rules. The DIFF column compares against the initial results in Table 6.

	PREC	RECALL	F-SCORE	DIFF
Original structure	88.78	88.86	88.82	−0.03
NML and JJP brackets only	75.27	58.33	65.73	−1.71
All brackets	88.51	88.07	88.29	−0.06

Once again, we were surprised to find that the F-score has been reduced, though by only a small amount overall, which chiefly comes from the NML and JJP brackets. This can be explained by considering an example NML: *lung cancer*. The corrected head-finding rule conditions the modifier *lung* on the head *cancer*. This NML constituent would then be quite likely, as the set of possible modifiers is restricted by the probability distribution. However, the reverse (conditioning the head *cancer* on the modifier *lung*) would also be informative, as the set of heads is likewise restricted. An NML's left-most token is rarely *the* or another uninformative token, and thus the uncorrected head-finding rules are also quite effective.

Furthermore, for NMLs such as *Judge Curry* or *Mr Vinken*, the left-most token is actually a much better generalization to pass up the tree and base probabilistic actions upon. Finally, Bikel (2004, §6.1.1) and Chiang and Bikel (2002) note that head-finding rules do not affect Collins's models to a large degree. Using a much simpler set of rules degrades performance by only a small amount, whereas an optimal set of rules derived using Expectation Maximization (EM) does not perform significantly better than the standard ones. For these reasons, choosing the left- or right-most token as the head achieves similar performance.

5.4 The Base-NP Submodel

The next alteration to the parser is to turn off the base-NP submodel. Collins (1999, page 179) explains that this separate model is used because the Penn Treebank does not fully annotate internal NP structure, something that we have now done. Hopefully, with these new brackets in place, we can remove the NP submodel and perhaps even improve performance in doing so.

We experimented with three different approaches to turning off the base-NP model. All three techniques involved editing the parser code:

1. Changing the isBaseNP() method to always return false. This means that the main model, rather than the NP submodel, is always used.
2. Removing the preprocessing step that creates NPB nodes. This alteration will have the same effect as the first change, and will also remove the distinction between NP and NPB nodes.
3. Changing the isNP() method to return true for NMLs. This will affect which NPs are turned into NPBs during the preprocessing step, as NPs that dominate NMLs will no longer be basal.

The third change does not turn the base-NP model off as such, but it does affect where it functions.

The results in Table 9 show that overall F-score has decreased in all cases. In the first change, to isBaseNP(), performance on only NML and JJP brackets has actually increased by 3.78% F-score, although the original structure is almost 10% worse. The second change, to the preprocessing step, results in a much smaller loss to the original structure, but also not as big an increase on the internal NP brackets. The third change, to isNP(), is most notable for the large drop in performance on the internal NP structure.

There are a few reasons for these results, which demonstrate the necessity of the base-NP submodel. Collins (1999, §8.2.2) explains why the distinction between NP and NPB nodes is needed: Otherwise, structures such as that in Figure 3, which *never* occur in the Treebank, are given too high a probability. The parser needs to know where NPs

Table 9
Performance achieved with the Bikel (2004) parser and the base-NP model off in three different ways: (1) No NP submodel, (2) No NPB nodes, (3) No NPB nodes when the NP is dominating a NML. DIFF is again comparing against the initial results in Table 6.

		PREC	RECALL	F-SCORE	DIFF
1	Original structure	72.09	88.19	79.33	−9.52
	NML and JJP brackets only	72.93	69.58	71.22	+3.78
	All brackets	72.11	87.71	79.14	−9.21
2	Original structure	87.75	87.65	87.70	−1.05
	NML and JJP brackets only	72.36	69.27	70.78	+3.34
	All brackets	87.37	87.17	87.27	−1.08
3	Original structure	86.90	88.66	87.77	−1.08
	NML and JJP brackets only	48.61	3.65	6.78	−60.66
	All brackets	86.83	86.46	86.64	−1.71

will not recurse anymore (when they are basal), so that it can generate the correct flat structure. Furthermore, the third change effectively treats NP and NML nodes as equivalent, and we have already seen problems caused by this approach in Section 5.2.

5.5 Bracket Structure

We have now seen how a Collins-style parser performs on internal NP structure, but the question remains about whether the structure itself is optimal. Treebank structure can have a large effect on parser performance, as has been studied by many researchers. Collins (2003, page 621) notes that binary trees would be a poor choice, as the parser loses some context sensitivity, and the distance measures (§3.1.1) become ineffective. He advocates one level of bracketing structure per X-bar level.

Goodman (1997) on the other hand, explicitly converts trees to a binary branching format as a preprocessing step, in order to avoid problems from varying structures. Johnson (1998) finds that the performance of simple PCFGs can be improved through tree transformations, whereas Klein and Manning (2001) observe that some simple tree transformations can increase parsing speed. Petrov et al. (2006) perform automatic tree transformations by splitting nonterminal symbols, creating a smaller grammar that achieves state-of-the-art performance. The variation shown in these approaches, all for the same task, highlights the difficulty in identifying optimal tree structure.

Kübler (2005) investigates two German treebanks with different annotation schemes, and finds that certain properties, such as having unary nodes and flatter clauses, increase performance. Rehbein and van Genabith (2007) suggest that evaluation

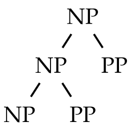


Figure 3
This structure, which never appears in the corpus, will be generated unless base-NPs are marked.

methods are also affected by treebank structure, showing that the Parseval measures are biased towards an increased number of non-terminal nodes.

It may be argued that a better representation for some NPs is to explicitly bracket right-branching structure. For example, in the NP `the New York Stock Exchange`, if there was a bracket around `New York Stock Exchange`, then it would be useful training for when the parser comes across `New York Stock Exchange composite trading` (which it does quite often). The parser should learn to add a bracket in both cases. The current bracketing guidelines do not mark right-branching constituents, they are simply assumed implicitly to be there.

We experiment with automatically adding these right-branching brackets and then examine what difference this change makes. Table 10 shows that overall performance drops by 1.51% F-score. This was a surprising result, as there are a number of easily recoverable brackets that are introduced by making right-branching structure explicit. For example, a POS tag sequence of `DT NN NN` is always right-branching. This explains the more than 10% increase in F-score when evaluating internal NP brackets only. As Rehbein and van Genabith (2007) found, increasing the number of non-terminal nodes has caused an increase in performance, though we may question, as they do, whether performance has truly increased, or whether the figure is simply inflated by the evaluation method. Either way, the deleterious effect on overall performance suggests that right-branching structure should be left implicit.

5.6 Test Set Results

Having found that the best performing model is the initial one with no alterations, we now evaluate its results on the test data: Section 23. Table 11 shows that, as with the Section 00 results, the original Penn Treebank structure is barely affected by the additional `NML` and `JJP` brackets. The new brackets themselves are recovered slightly better than they were on the development data, achieving a figure that is almost the same as the suggestion baseline in this case.

These results confirm those we saw in our initial experiments: Recovering NP structure is a difficult task for the Collins (2003) model. As a result, there is a slight drop in overall performance.

5.7 Error Analysis

Despite the large number of experiments we have performed in this section, we are no closer to outperforming the suggestion baseline established in Section 4.3. The highest accuracy has come from the unaltered parser, and changes to the corpus and model have proven unsuccessful. We need to look at the errors being made by the parser, so that any problems that appear can be solved. Accordingly, we categorized each of the

Table 10
Performance achieved with the Bikel (2004) parser and explicit right-branching structure. The DIFF column compares against the initial results in Table 6.

	PREC	RECALL	F-SCORE	DIFF
Original structure	87.96	88.06	88.01	−0.84
`NML` and `JJP` brackets only	82.33	74.28	78.10	+10.66
All brackets	87.33	86.36	86.84	−1.51

Table 11
Performance achieved with the Bikel (2004) parser, final results on the test set. The suggestion baseline is comparable to the NML and JJP brackets only figures, as are the Original PTB and Original structure figures.

	PREC.	RECALL	F-SCORE
Original PTB	88.58	88.45	88.52
Suggestion baseline	94.29	56.81	70.90
Original structure	88.49	88.53	88.51
`NML` and `JJP` brackets only	80.06	63.70	70.95
All brackets	88.30	87.80	88.05

560 `NML` and `JJP` errors in our initial model through manual inspection. The results of this analysis (performed on the development set) are shown in Table 12, together with examples of the errors being made. Only relevant brackets and labels are shown in the examples; the final column describes whether or not the bracketing shown is correct.

Table 12
Error analysis for the Bikel (2004) parser on the development set, showing how many times the error occurred (#), the percentage of total errors (%), and how many of the errors were false positives (FP) or false negatives (FN). If a cross (×) is in the final column then the example shows the error being made. On the other hand, if the example is marked with a tick (√) then it is demonstrating the correct bracketing.

ERROR	#	%	FP	FN	EXAMPLE	
Modifier attachment	213	38.04	56	157		
`NML`	122	21.79	21	101	*lung cancer deaths*	×
Entity structure	43	7.68	24	19	*(Circulation Credit) Plan*	×
Appositive title	29	5.18	6	23	*(Republican Rep.) Jim Courter*	√
`JJP`	10	1.79	4	6	*(More common) chrysotile fibers*	√
Company/name	9	1.61	1	8	*(Kawasaki Heavy Industries) Ltd.*	√
Mislabeling	92	16.43	30	62	*(ADJP more influential) role*	√
Coordinations	92	16.43	38	54	*(cotton and acetate) fibers*	√
Company names	10	1.79	0	10	*(F.H. Faulding) & (Co.)*	√
Possessives	61	10.89	0	61	*(South Korea) 's*	√
Speech marks/brackets	35	6.25	0	35	*(" closed-end ")*	√
Clear errors	45	8.04	45	0		
Right-branching	27	4.82	27	0	*(NP (NML Kelli Green))*	×
Unary	13	2.32	13	0	*a (NML cash) transaction*	×
Coordination	5	0.89	5	0	*(NP a (NML savings and loan))*	×
Structural	8	1.43	3	5	*(NP . . . spending) (VP (VBZ figures) . . .)*	×
Other	14	2.50	8	6		
Total	560	100.00	180	380		

The most common error caused by an incorrect bracketing results in a modifier being attached to the wrong head. In the example in the table, because there is no bracket around *lung cancer*, there is a dependency between *lung* and *deaths*, instead of *lung* and *cancer*. The example is thus incorrectly bracketed, as shown by the cross in the final column of the table. We can further divide these errors into general NML and JJP cases, and instances where the error occurs inside a company name or in a person's title.

The reason for these errors is that the *n*-grams that need to be bracketed simply do not exist in the training data. Looking for each of the 142 unique *n*-grams that were not bracketed, we find that 93 of them do not occur in Sections 02–21 at all. A further 17 of the *n*-grams do occur, but not as constituents, which would make reaching the correct decision even more difficult for the parser. In order to fix these problems, it appears that an outside source of information must be consulted, as the lexical information is currently not available.

The next largest source of errors is mislabeling the bracket itself. In particular, distinguishing between using NP and NML labels, as well as ADJP and JJP, accounts for 75 of the 92 errors. This is not surprising, as we noted during the final preparation of the corpus (see Section 3.3) that the labels of some NPs were inconsistent. The previous relabeling experiment suggests that we should not evaluate the pairs of labels equally, meaning that the best way to fix these errors would be to change the training data itself. This would require alterations to the original Penn Treebank brackets, something we avoided during the annotation process. In this case the example shown in the table is correct (with a tick in the final column), while the parser would've incorrectly labeled the bracket as JJP.

Coordinations are another significant source of errors, because coordinating multi-token constituents requires brackets around each of the constituents, as well as a further bracket around the entire coordination. Getting just a single decision wrong can mean that a number of these brackets are in error. Another notable category of errors arises from possessive NPs, which always have a bracket placed around the possessor in our annotation scheme. The parser is not very good at replicating this pattern, perhaps because these constituents would usually not be bracketed if it were not for the possessive. In particular, NML nodes that begin with a determiner are quite rare, only occurring when a possessive follows. The parser also has difficulty in replicating the constituents around speech marks and brackets. We suspect that this is due to the fact that Collins's model does not generate punctuation as it does other constituents.

There are a number of NML and JJP brackets in the parser's output that are clearly incorrect, either because they define right-branching structure (which we leave implicit) or because they dominate only a single token. The only single token NMLs exist in coordinations, but unfortunately the parser is too liberal with this rule. The final major group of errors are structural; that is, the entire parse for the sentence is malformed, as in the example where *figures* is actually a noun.

From this analysis, we can say that the modifier attachment problem is the best to pursue. Not only is it the largest cause of errors, but there is an obvious way to reduce the problem: Find and make use of more data. One way to do this with a Collins-style parser would be to add a new probability distribution to the model, akin to subcategorization frames and distance measures (Collins 2003, §3.2 and §3.1.1). However, doing so would be a challenging task.

We take a different approach in Section 6: using a separate NP Bracketer. This allows us to use *n*-gram counts as well as a wide range of other features drawn from many different sources. These can then be included in a model specifically dedicated to parsing NP structure. This approach is similar to the machine translation system of Koehn (2003), which uses a parser to identify NPs and then translates them using an NP-specific subsystem. Additional features are included in the subsystem's model, improving accuracy from 53.9% to 67.1%. When this subsystem is embedded in a word-based MT system, its BLEU score (Papineni et al. 2002) increases from 0.172 to 0.198.

6. Noun Phrase Bracketing

In this section, we will use NP bracketing techniques (as described in Section 2.3) to improve upon the parser's performance of only 67.44%.

We divide NPs into two categories:[5]

Simple NPs: are exactly three words long and contain only nouns.

Complex NPs: include all simple NPs as well as those that are longer than three words, contain non-nominal parts of speech (such as adjective, determiner, etc.), and include clausal modifiers. We consider every constituent annotated as NP in the Penn Treebank to be a complex NP.

We will begin with simple NPs, as most previous work has limited itself in these ways and we wish to allow a comparison between our experiments and those in the literature. Our eventual aim, however, will be to build a post-processor that can bracket the full range of NPs in the Penn Treebank. Thus, in our later experiments we will take on the much more difficult task of bracketing complex NPs from our newly annotated corpus.

6.1 Data

In order to build an NP Bracketing system, there must be data to evaluate it, and for supervised models, to train on as well. We extract both a simple and a complex NP data set from our extended Penn Treebank.

6.1.1 Simple NPs. Simple NPs are extracted from our extended Penn Treebank data as follows. If the last three children of an NP are nouns, then they became an example in our data set. We mark the NP as left-branching if the first and second words are bracketed, and as right-branching otherwise. This method (assuming that all NPs have a right-most head) will retrieve all possible simple NPs from the corpus. It also means that we will not have headless sequences of modifiers in the data set, because only the right-most part of the NP is being looked at. This also allows us to retrieve simple NPs from NPs longer than three words, by simply ignoring the left-most modifier(s). One final step is to remove examples where each word has the same NER tag, ignoring many flat base-NP cases such as *John A. Smith*. Lauer (1995) used a similar approach to collect three noun sequences from Grolier's encyclopedia.

Some example NPs from the Penn Treebank are shown below:

```
(NP (NN executive) (NN vice) (NN president) )
(NP (NML (NN lung) (NN cancer) ) (NNS deaths) )
(NP (JJ separate) (NN board) (NNS meetings) )
(NP (DT an) (NN assistant) (NN state) (NN attorney) (NN general) )
(NP (NML (NNP New) (NNP York) ) (NNP Stock) (NNP Exchange) )
```

[5] Note that this is our own terminology, and has no relation to other uses of these terms in the literature.

From this set we extract one left-branching NP (*lung cancer deaths*), and two right-branching NPs (*executive vice president* and *state attorney general*). Other potential sequences that we don't extract include: *separate board meetings*, as *separate* is an adjective; *assistant state attorney*, as these tokens are not right-most in the NP; and *York Stock Exchange* as these tokens are not (even implicitly) dominated by single node.

This process results in 5,569 three-word NPs for bracketing, which is an order of magnitude larger than all previous data sets. Previous researchers have typically used Lauer's set (244 NPs) or created their own small set (∼500 NPs at most). This new, much larger data set means that we can carry out large-scale machine learning effectively, rather than using unsupervised methods.

Statistics comparing our new data set to those used by other researchers are shown in Table 13. As can be seen, the Penn Treebank-based corpus is significantly larger than all other data sets. The distribution of left- and right-branching NPs also appears to vary greatly, which may be affected by the content of the corpus. The Nakov and Hearst (2005) biomedical and Barker (1998) small engines data sets are both very technical texts, and the Buckeridge and Sutcliffe (2002) AmiPro software manual and Buckeridge and Sutcliffe (2002) Time magazine articles are probably aimed at a more general audience.

6.1.2 Complex NPs. We have also extracted another even larger data set of complex NPs for the experiments in Section 6.4. For this set we retrieve an example for each NP of length three or more in the Penn Treebank. We will only be identifying the structure in NML and JJP brackets and so NPs with other child nodes (e.g., prepositional phrases) must be treated specially. We choose a simple approach, taking the head word of the child node to represent the entire constituent. This means we can treat such NPs in the same manner as base-NPs and don't have to store internal structure that our system will not attempt to parse. An example of this is shown in the first row of Table 14, where *ABC Co* and *the market leader* would both have been NP nodes. We have elided the words in square brackets, leaving only the heads of those constituents.

Some common POS tag sequences (e.g., initial determiner and final possessive) are unambiguous in three word NPs, and so we remove these cases. This has the side effect of increasing the ambiguity in the data and making the task harder on average. This leaves 53,568 instances in our data set, which is two orders of magnitude larger than any that has been created previously.

Table 14 shows the most common POS tag sequences in our complex NP data set. The entropy of the distribution of bracketings for the POS tag sequences gives an indication

Table 13
Comparison showing the sizes of various NP bracketing corpora.

CORPUS	# ITEMS	LEFT (%)	RIGHT (%)
Penn Treebank	5,569	59	41
Lauer (1995)	244	67	33
Buckeridge and Sutcliffe (2002) AmiPro	307	58	42
Buckeridge and Sutcliffe (2002) CISI	235	63	37
Buckeridge and Sutcliffe (2002) CRAN	223	74	26
Buckeridge and Sutcliffe (2002) Time	214	48	52
Nakov and Hearst (2005) Biomedical	430	84	16
Barker (1998) SPARC	188	45	55
Barker (1998) small engines	164	91	9

Table 14
The most common bracketings of POS tag sequences in our complex NP corpus.

#	ENTROPY	SEQUENCE	EXAMPLE
2,228	0.00	`( NNP , NN )`	*[ABC] Co , [the market] leader*
1,796	1.00	`( ( NNP NNP ) NNP )`	*John Smith Co.*
1,762	1.00	`( NNP NNP NNP )`	*John A. Smith*
1,481	0.54	`( JJ NN NNS )`	*high interest rates*
1,359	0.59	`( DT JJ NN NN )`	*the high interest rate*
1,054	0.13	`( JJ JJ NNS )`	*big red cars*

of the difficulty of the task. Larger entropy means that the sequence is more ambiguous, because there are many bracketing alternatives to choose from and/or because the alternatives are close to equally likely. The entropy figure for the `NNP NNP NNP` bracketings reinforce the result we saw in Section 5.7: A sequence of three nouns is very hard to bracket, as there is no good baseline decision.

Figure 4 shows a histogram of the entropy distribution across POS tag sequences. Although 43.71% of all sequences have a single bracketing, the majority of sequences are ambiguous. There is a spike just below an entropy of 1, mostly made up of sequences with two almost equally likely bracketings. This demonstrates that complex NP bracketing is far from a trivial task.

6.2 Unsupervised Experiments

With our new data set of simple NPs, we began running experiments similar to those carried out in the literature (Nakov and Hearst 2005). Refer back to Section 2.3 for a reminder of the models typically used for this task. We implemented both an adjacency and dependency model, and three different association measures: the raw bigram count, the bigram probability, and χ^2.

$$\text{Raw bigram count} = count(w_i, w_j) \tag{7}$$

$$P(w_i|w_j) = \frac{count(w_i, w_j)}{count(w_j)} \tag{8}$$

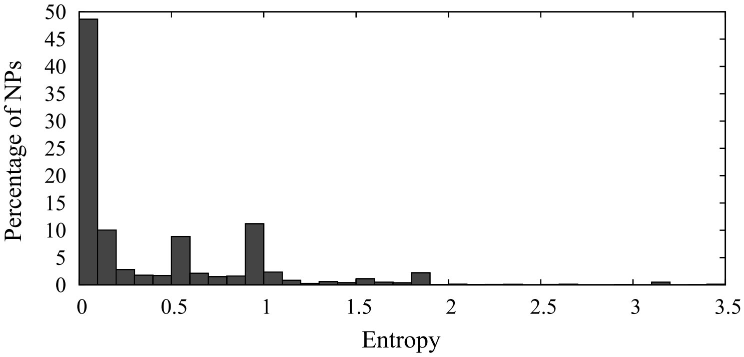


Figure 4
Entropy over the bracketings of POS tag sequences from the complex NP corpus.

$$\chi^2(w_i, w_j) = \frac{N(AD - BC)^2}{(A + C)(B + D)(A + B)(C + D)} \tag{9}$$

$$\text{where } A = count(w_i, w_j) \tag{10}$$

$$B = count(w_i, \bar{w}_j) \tag{11}$$

$$C = count(\bar{w}_i, w_j) \tag{12}$$

$$D = count(\bar{w}_i, \bar{w}_j) \tag{13}$$

$$\text{and } N = A + B + C + D \tag{14}$$

$\bar{w}$ indicates any word *except* w

Our counts come from three different sources: Google and MSN search engine hit counts, and from the Google Web 1T corpus (Brants and Franz 2006), which contains n-gram counts collected from 1 trillion words of Web text. We can calculate N, for the χ^2 measure shown in Equation (9), as the total number of bigrams in the Web 1T corpus (910,884,463,583), and take the same estimate as Nakov and Hearst of 8 trillion when using search engine counts.

One problem with the bigram probability and χ^2 measures is that a single zero count will cause the entire measure to be zero, ignoring the effect of other non-zero counts. To solve this problem, Nakov and Hearst (2005) apply a basic form of smoothing: adding 0.5 to each frequency count. Although this is not a particularly effective form of smoothing, we take a similar approach so that our results will be comparable with theirs.

The results from the experiments, on both Lauer's and our data set, are shown in Tables 15 and 16, respectively. Our results on Lauer's corpus are similar to those reported previously, with the dependency model outperforming the adjacency model on all measures. The Web 1T counts are the most effective, and the raw counts—the

Table 15
Unsupervised results for the simple NPs in Lauer's data set.

COUNTS	ADJACENCY			DEPENDENCY		
	RAW	PROB.	χ^2	RAW	PROB.	χ^2
Google	72.5	68.4	**73.0**	**77.5**	75.0	76.2
MSN	71.3	65.6	**72.1**	**75.0**	74.6	74.6
Web 1T	74.2	70.5	**75.4**	81.2	**82.8**	77.5

Table 16
Unsupervised results for the simple NPs in the Penn Treebank data set.

COUNTS	ADJACENCY			DEPENDENCY		
	RAW	PROB.	χ^2	RAW	PROB.	χ^2
Google	75.53	69.85	**79.98**	69.58	68.61	**69.94**
MSN	76.53	74.38	**80.07**	69.22	69.29	**69.82**
Web 1T	**80.05**	79.62	79.33	74.18	**75.18**	70.71

simplest association measure—work surprisingly well. The results on the new corpus are also surprising, as the adjacency model outperforms the dependency model by a wide margin. Once again, the Web 1T counts perform well in all cases, although the best result is from the MSN search engine. The χ^2 measure gives the highest accuracy for both search engines, but is least effective with the Web 1T counts. The two search engines give reasonably similar results on both data sets.

Our analysis shows that the good performance of the adjacency model comes from the large number of named entities in the corpus. When we remove all items that have any word as a named entity, the results are reversed, and the dependency model is superior. On the 1,556 NPs that remain, using Web 1T counts and the χ^2 measure, the adjacency model achieves 71.85% accuracy, and the dependency model attains 73.84%. The other count sources and association measures show the same trend.

6.2.1 n-gram Variations. Both the adjacency and dependency models are relatively knowledge-poor, only utilizing a pair of bigram counts in order to make a decision. In order to increase the amount of information available, we retrieved hit counts for a number of other variations on the simple bigrams, as proposed by Nakov and Hearst (2005). For example, the bigram joined by a hyphen to form a single token, or with a possessive marker. The full list is shown in Table 17 — Google (G), MSN (M), Web 1T (W), and snippets (S). Also shown is whether or not the count source used that pattern.

Table 17
Variations on the basic query used in our experiments. The final four columns show which count sources the variation was used with: Google, MSN, Web 1T, and/or Snippets. A tick (√) indicates that the count source was used, and a cross (×) means that it was not.

NAME	LEFT BRANCHING	RIGHT BRANCHING	G	M	W	S
Wildcard 1	brain stem * cells	brain * stem cells	√	×	×	×
Wildcard 2	brain stem * * cells	brain * * stem cells	√	×	×	×
Wildcard 3	brain stem * * * cells	brain * * * stem cells	√	×	×	×
Reverse wildcard 1	cells * brain stem	stem cells * brain	√	×	×	×
Reverse wildcard 2	cells * * brain stem	stem cells * * brain	√	×	×	×
Reverse wildcard 3	cells * * * brain stem	stem cells * * * brain	√	×	×	×
Adjacency concat.	brainstem	stemcells	√	√	√	×
Dependency concat.	brainstem	braincells	√	√	√	×
Concatenation triple	brainstem cells	brain stemcells	√	√	√	×
Swap first two words	brain stem cells	stem brain cells	√	√	√	×
Reorder	cells brain stem	stem cells brain	√	√	√	×
Abbreviation	brain stem bs cells	brain stem cells sc	√	√	√	×
Abbreviation w/brackets	brain stem (BS)	stem cells (SC)	×	×	√	×
Possessive	stem's	brain's	√	√	×	×
Possessive triple	brain stem's cells	brain's stem cells	√	√	√	√
Capitalization	brain stem Cells	brain Stem cells	×	×	√	√
Internal hyphenation	brain-stem cells	brain stem-cells	×	×	√	√
External hyphenation	brain stem cells-*	*-brain stem cells	×	×	√	√
Internal slash	brain/stem cells	brain/stem cells	×	×	√	√
External slash	brain stem cells/*	*/brain stem cells	×	×	√	√
Left brackets	(brain stem) cells	(brain) stem-cells	×	×	√	√
Right brackets	brain stem (cells)	brain (stem-cells)	×	×	√	√
Comma	brain stem, cells	brain, stem cells	×	×	√	√
Colon	brain stem: cells	brain: stem cells	×	×	√	√
Period	brain stem. cells	brain. stem cells	×	×	√	×
N&H period	brain. stem cells	brain stem. cells	×	×	√	√

Some patterns cannot be used by some count sources, for example, MSN does not do wildcards searches, and Web 1T only goes up to 5-grams. Snippets is another source of counts suggested by Nakov and Hearst (2005), utilizing the short piece of text that comes with each search result. These snippets come from the Google search engine.

Nakov and Hearst (2005) used these *n*-gram variations in a complicated voting scheme, where different counts from different sources were given hand-tuned weights and then combined. Rather than implementing such a complex algorithm, we performed some simpler voting experiments. Each *n*-gram variation was given a single unweighted vote. If the left and right counts were equal, then the variation supplied no vote, and if the final votes were equally split, then we defaulted to left-branching.

We performed a greedy search through the possible sets of voters, to optimize performance on Lauer's data. Our best result uses the voters in Table 18. This set achieves 90.2% accuracy, a similar figure to Nakov and Hearst's 89.3%, without the morphological or paraphrase queries, and without manually weighting any features.

Both of these voting systems are effectively supervised models, however, where the training process determines the optimal set of features (and weights for Nakov and Hearst's model). As such, a separate training set should be used to avoid overestimating performance. Due to the small size of Lauer's data set, we followed Nakov and Hearst (2005) in developing the test data itself. They note that Lapata and Keller (2004) divided Lauer's in half to develop, and that the difference in performance on the two halves was negligible. Despite this, we argue that neither of the results give an accurate representation of NP Bracketing performance. The optimal set of voters we identified is unlikely to be as effective for any other data set.

We can test this by applying the Lauer optimal voter set (from Table 18) to the Penn Treebank data. This results in 76.49% accuracy, which is lower than using the adjacency model alone. Considering the seemingly random selection of voters, this is not particularly surprising, although it may be because of the different performance levels of the dependency and adjacency models of the two corpora. In the following section, we will perform the reverse experiment, training on the Penn Treebank data and testing on Lauer's. This will provide a better idea of the true performance levels.

The main problem with a voting technique is that it does not effectively combine competing factors into a single model. The new Penn Treebank data set enables a much better solution: Apply a robust supervised model. This Penn Treebank data set is an order of magnitude larger than Lauer's, making available a sufficient amount of training, development, and test data for the first time.

Table 18
The optimal set of voters for the simple NPs in Lauer's data set, as found by our greedy search method.

GOOGLE	WEB 1T	SNIPPETS
Wildcard 2 Abbreviation Possessive	Dependency probability Concatenation triple Abbreviation with brackets Capitalization Internal hyphenation Internal slash External slash Left brackets Right brackets	Possessive Capitalization Internal hyphenation Right brackets

6.3 Supervised Models

Supervised models typically outperform unsupervised models for most NLP tasks. For NP bracketing, the small quantity of gold-standard data has meant that few supervised models have been implemented, and those that have been performed poorly. With our new, significantly larger data set covering the Penn Treebank, we have built the first large-scale supervised NP bracketer.

The data set is split into training, development, and test sets, with 4,451; 559; and 559 NPs, respectively. We use the MegaM Maximum Entropy classifier (Daumé III 2004), and discretize non-binary features using Fayyad and Irani's (1993) algorithm. Maximum entropy models allow diverse and overlapping features to be incorporated in a principled manner. Our initial features use counts from Google, Web 1T, and the snippets. We no longer use MSN because it produces similar results to Google. We use the adjacency and dependency models, and all three association measures. The *n*-gram variations in Table 17 for the three count sources are also used, but only the raw count. This is because the counts are often too small. For each of these, there is one feature for the left association measure, another for the right association measure, and a pair of binary features representing whether the left or right measure is greater. If the two measures are equal, then neither binary feature is on.

The results on our Penn Treebank development set are shown in Table 19, compared to an unsupervised adjacency model and the unsupervised voting system from Section 6.2.1. As we described there, calling the latter model unsupervised is a misnomer, as the set of voters needs to be optimized on training data. With the larger Penn Treebank corpus available, we can now "train" this unsupervised voting model on the training set, rather than on the test set itself. This avoids over-estimating its performance figures.

Table 19
Comparing unsupervised approaches to a supervised model on the development data of the Penn Treebank simple NP corpus. The last two results groups show a subtractive analysis, removing individual feature groups from the All features model.

MODEL	F-SCORE
Unsupervised, Web 1T adjacency	82.5
Unsupervised, voting	89.6
All unsupervised features	90.2
All supervised features	89.5
All features	93.0
−Google	93.0
−Snippets	93.0
−Web 1T corpus	92.1
−Lexical	92.3
−POS	92.5
−NER	92.1
−Context sentence	92.7
−Context window	92.0
−Semantic	**93.8**

The supervised model outperforms the unsupervised voting model by 0.6%, even though both models are using the same information to base their decisions on. This improvement comes from the supervised model's ability to weight the individual contributions of all the unsupervised counts from Google and the Web 1T corpus.

We can also test on Lauer's data set using the supervised model trained on Penn Treebank data. The result is an 82.4% accuracy figure, which is higher than our unsupervised dependency model and Lauer's. However, it is much lower than Nakov and Hearst's (2005) best result and our own voting scheme. This suggests that the voting schemes, by training on their own test data, have over-estimated their performance by about 9%.

6.3.1 Additional Features. One of the main advantages of using a maximum entropy classifier is that we can easily incorporate a wide range of features in the model. We now add lexical features for all unigrams, bigrams, and the trigram within the NP. All of these features are labeled with the position of the *n*-gram within the NP.

Because we are bracketing NPs in situ rather than stand-alone NPs (like Lauer), the context around the NP can be exploited as well. To do this we added bag-of-word features for all words in the surrounding sentence, and well as specific features for a two-word window around the NP. For the context sentence, there are features for words before the NP, after the NP, and either before or after the NP.

We have access to gold-standard POS and NER tags, from the Penn Treebank and the BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein 2005), respectively. These are used by adding generalized *n*-gram and context window features, replacing the words with their POS and NER tags. POS tags are included even though all the words in the NP are nouns for these simple NP experiments, as they may be proper and/or plural. We use the coarse-grained NER tags, including the `B−` and `I−`.

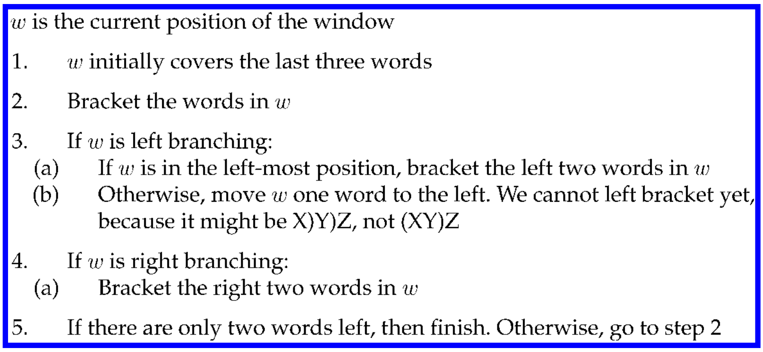
Finally, we incorporate semantic information from WordNet (Fellbaum 1998). These features are intended to work similarly to how Lauer (1995) groups nouns into semantic classes. For each sense of each word in the NP, we extract a semantic feature for its synset, and also the synset of each of its hypernyms up to the WordNet root. These features are marked with how far up the tree from the original synset the hypernym is, but there is also an unordered bag-of-hypernyms for all senses.

Table 19 shows the results for a model using only the supervised features, and a combination of the supervised and unsupervised features. It also presents a subtractive analysis. The unsupervised features alone outperform the supervised ones by themselves, and using both together gives a further increase. The Google and snippets features do not appear to contribute at all, probably because they overlap significantly with the Web 1T searches. Of the supervised features, the context window and NER are most important but all make a positive contribution, except for the semantic features. One reason why the semantic features perform negatively is that we have not attempted to disambiguate between word senses. We do not have enough data for the model to accurately choose which senses are accurate, and furthermore, many of the synsets used are close to the WordNet root and thus uninformative. Our best performance of 93.8% F-score is obtained by removing this group.

Finally, results on the test set are shown in Table 20. The supervised model has improved over the unsupervised baseline by 6.6%, demonstrating that the voting method's performance is quite variable, whereas the Maximum Entropy model remains consistent.

Table 20
Test set results for the Penn Treebank simple NP corpus comparing the best supervised and unsupervised models.

MODEL	ACCURACY
Unsupervised, Web 1T adjacency	77.6
Unsupervised, voting	86.8
Best supervised model	93.4



```
w is the current position of the window
1.    w initially covers the last three words
2.    Bracket the words in w
3.    If w is left branching:
  (a)    If w is in the left-most position, bracket the left two words in w
  (b)    Otherwise, move w one word to the left. We cannot left bracket yet,
         because it might be X)Y)Z, not (XY)Z
4.    If w is right branching:
  (a)    Bracket the right two words in w
5.    If there are only two words left, then finish. Otherwise, go to step 2
```

Figure 5
Barker's (1998) NP bracketing algorithm.

6.4 Complex NPs

All of our experiments so far (and almost all results from the literature) have only focused on NPs that consist of exactly three nouns (noun compound bracketing). This is a simplification of the actual problem, where longer NPs with higher levels of ambiguity make finding the correct bracketing significantly harder. Adjectives, determiners, and other non-nominal parts of speech also complicate the task.

Barker (1998) describes a method for bracketing these complex NPs, by reducing the problem to a series of three word bracketing decisions using a sliding window. These decisions can then be made using the techniques described previously for simple[6] NPs. Barker's algorithm is shown in Figure 5.

When a pair of words is bracketed, the head is chosen to represent the phrase and remains in the window. We use the standard head-finding rules of Collins (1999). The window then expands one word to the right, unless it is already right-most in which case it grows to the left.

For these experiments, we use the complex NP data set previously described in Section 6.1. The 53,568 complex NPs are split in a 8:1:1 ratio, giving 42,854 examples for training and 5,357 for development and testing.

[6] We abuse the terminology slightly here, as these NPs can include non-noun parts of speech.

6.4.1 Evaluation Measures. The complex NP results are evaluated using several measures. Firstly, matching brackets is the standard Parseval evaluation method (Black et al. 1991). Secondly, because our annotation only marks left-branching structure explicitly (see Section 3), we can also report implicit matching brackets, where we automatically insert the implicit right-branching brackets for evaluation purposes. This takes into account fully right-branching NPs, which contribute no score using the harsher, explicit matching brackets evaluation. For example, a baseline of always choosing right branching will score 0.0, as no explicit brackets are needed.

We also measure exact NP match, which measures the percentage of complex NPs that are entirely correct, and the model's performance on the three word NPs that are processed during Barker's algorithm. We only report accuracy for implicit brackets, as there is a set number of brackets dependent on the length of the word, and so precision and recall are always equal. Finally, note that the three-word NPs are different for each model, as the next three word NP to bracket depends on the decisions made previously for this complex NP. Consequently, the numbers for this measure are not directly comparable.

6.4.2 Complex NP Results. Our first experiment implements Barker's algorithm, using only the χ^2 dependency and adjacency methods to make each decision. We only use counts from the Web 1T corpus, because performing Web searches has become impractical with the increased data set size and NP length. The difficulty of complex NP bracketing can be seen in Table 21, by the drop in performance using these simple approaches (e.g., from 79.33% in Table 16 to to 56.29% for the adjacency model).

We next apply our supervised approach to complex NPs. This is more complicated now as we need to extract a training set of three word windows from the complex NPs. To do this, we run Barker's algorithm on the 42,854 complex NPs. At each decision point, we bracket left or right according to the gold standard, and store the three-word window as a training example. This process is similar to a shift-reduce parser, such as that used in the RASP parser (Briscoe and Carroll 2006) or Ratnaparkhi's (1997) maximum entropy parsing model. The complex NP data produces 95,964 training examples.

We experiment with the same features used for simple NPs, as well as some novel features. Firstly, we add features encoding the non-head words when the window already contains a bracket. This means that for each bracket that has already been inserted for the complex NP, all words dominated by the bracket are labeled with their position in the window and added as features. For example, consider the NP *French onion soup bowl* after *onion soup* has been bracketed. Although only *soup* remains in the window, *onion* is added as a feature and labeled as the first word of the second node in the window. The POS tag, NER tag, and Web 1T count of these words are also included as separate features. This feature group proved to be very informative for the model.

Secondly, we add the bigram of the words on the NP border, that is, where it overlaps with the context. Thirdly, we measured the entropy of every POS tag sequence in the training data. Some of these figures were shown earlier in Table 14. Those sequences with entropy below 0.05 bits, that is, the ones that are quite unambiguous, were then extracted. For example, DT JJ NN is almost always right-branching. We then implemented a feature explicitly encoding their most common branching. There are only two features for left- and right-branching, rather than features for each POS tag.

Finally, we introduce features based on the Bikel (2004) parser's output, which have been informative in PCFG parsing. For the parent and grandparent of the NP, we add a feature for the phrase label, the head-word and its POS tag, NER tag, and Web 1T count.

Table 21
Results with gold-standard complex NPs over four evaluation measures: (1) matched brackets (precision, recall, and F-score), (2) accuracy after including all implicit right-branching brackets, (3) exact NP match, and (4) accuracy over the simple NPs that were bracketed. The third group of results shows a subtractive analysis, removing individual feature groups from the All features model. The negative feature groups are removed for the Best results. The final three rows are calculated over the test set, rather than the development set as in all earlier experiments.

MODEL	MATCHED BRACKETS			IMPLICIT	EXACT	SIMPLE
	P	*R*	*F*			
Baseline – right branching	0.00	0.00	0.00	74.67	69.31	74.82
χ^2 Dependency	13.79	42.84	20.87	38.40	24.32	48.11
χ^2 Adjacency	16.13	41.00	23.15	49.27	34.50	56.29
All features	89.14	84.26	86.63	94.96	92.18	95.67
−Web 1T corpus	89.58	82.79	86.05	94.75	91.69	95.55
−Lexical	87.95	83.00	85.40	94.57	91.58	95.30
−POS	89.09	83.37	86.13	94.73	91.92	95.44
−NER	89.27	84.11	86.61	94.88	92.25	95.64
−Context sentence	91.45	86.11	88.70	95.69	93.19	96.33
−Context window	90.41	85.37	87.82	95.32	92.79	96.00
−Semantic	89.61	84.00	86.72	94.97	92.14	95.67
−Non-head words	84.84	81.58	83.18	94.03	90.76	94.80
−Border words	89.69	84.74	87.14	95.18	92.48	95.85
−POS tag sequence	89.93	85.05	87.42	95.26	92.70	95.96
−Parser	89.35	84.32	86.76	95.04	92.25	95.78
Best	92.09	86.37	89.14	95.88	93.49	96.48
Test – Baseline	0.00	0.00	0.00	72.79	68.08	72.98
Test – χ^2 Adjacency	17.76	41.89	24.95	50.08	36.64	57.07
Test – Best	91.32	88.19	89.73	95.95	93.69	96.68

The results are shown in Table 21. The supervised methods significantly outperform the unsupervised methods, with a matched brackets F-score comparable to the Bikel (2004) parser's overall performance. We carry out a subtractive analysis of the feature types, and find that both context feature groups, as well as the semantic, border, POS tag rule, and parser features all have a negative impact on performance.

Our optimal result comes from removing these feature groups. The 89.14% F-score achieved with this model is shown as **Best** in Table 21. All experiments were run using 500 iterations in MegaM, to allow the estimation to converge.

Finally, we applied our best model to the test data. The results, again in Table 21, are similar to those we achieved on the development set. This demonstrates that our complex NP Bracketing system achieves high performance on a wide range of inputs.

6.5 Parser Post-Processor

This final set of experiments uses the complex NP models as a post-processing step for a parser. As we saw previously in Section 5, the parser failed to outperform the suggestion baseline of 73.12% on NML and JJP brackets. We intend to surpass this figure with our NP bracketing technique, as it includes many additional feature types. This will be made more difficult by the fact that the post-processor is dependent on NPs identified by the parser, which are incorrect in approximately 10% of cases.

Atterer and Schütze (2007) use a similar approach, applying prepositional phrase attachment techniques to parser output, rather than to manually prepared, gold-standard examples. Doing so provides a more realistic view of a PP attachment system's performance, as it must contend with the additional difficulties created by parser error. The same applies to our NP Bracketing system.

We train the complex NP bracketer on gold-standard NPs from Sections 02–21, extracting 78,757 complex NPs that produce 132,195 three word training examples. The development set is created by first parsing Section 00 using the Bikel (2004) parser. We then extract the base NPs that the parser identifies and insert the gold-standard NP Bracketing for evaluation. We reject brackets that cross an NP boundary (i.e., a parsing error). This results in a development set of 3,946 complex NPs. A test set of 4,834 NPs is also produced in the same way from Section 23.

The results of these experiments, including subtractive analysis on the feature types, are shown in Table 22. Unfortunately, we find that many of the features are not helpful, and our best model utilizes only the Web 1T, lexical, POS, and non-head word features. Note that these are the same features that proved helpful in the subtractive analysis in the previous experiment with gold-standard data. This model achieves 82.10% matched bracket F-score.

Table 22
Results with Bikel (2004) parsed complex NPs over four evaluation measures: (1) matched brackets (precision, recall, and F-score), (2) accuracy after including all implicit right-branching brackets, (3) exact NP match, and (4) accuracy over the simple NPs that were bracketed. The third group of results shows a subtractive analysis, removing individual feature groups from the All features model. The negative feature groups are removed for the Best results. The final three rows are calculated over the test set, rather than the development set as in all earlier experiments.

MODEL	MATCHED BRACKETS			IMPLICIT	EXACT	SIMPLE
	P	*R*	*F*			
Baseline – right-branching	0.00	0.00	0.00	81.83	80.31	81.86
χ^2 Dependency	9.93	39.90	15.90	36.46	31.20	43.23
χ^2 Adjacency	12.50	42.55	19.32	47.24	41.41	51.37
All features	76.37	83.53	79.79	93.04	92.42	93.70
−Web 1T corpus	77.10	80.53	78.78	92.90	92.45	93.55
−Lexical	73.67	81.73	77.49	92.23	91.66	92.97
−POS	76.61	83.05	79.70	93.36	92.65	93.96
−NER	76.78	85.46	80.89	93.43	92.70	94.08
−Context sentence	78.53	84.86	81.57	93.78	93.26	94.38
−Context window	76.41	84.50	80.25	93.33	92.57	93.98
−Semantic	75.73	83.65	79.50	93.11	92.37	93.83
−Non-head words	74.21	81.97	77.90	93.15	91.97	93.78
−Border words	76.37	83.53	79.79	93.27	92.50	94.01
−POS tag sequence	76.77	84.62	80.50	93.55	92.68	94.27
−Parser	76.33	84.50	80.21	93.33	92.60	93.95
Best	78.78	85.70	82.10	94.08	93.41	94.67
Test – Baseline	0.00	0.00	0.00	79.68	79.56	79.68
Test – χ^2 Adjacency	14.55	44.06	21.87	48.83	42.22	53.44
Test – Best	81.16	87.08	84.02	94.22	93.94	94.78

Table 23
Performance comparison of suggestion baseline, parser, and the NP bracketer post-processing the parser's output on development data.

EVALUATING	MODEL	*P*	*R*	*F*
NML JJP	Suggestions	95.64	59.36	73.25
	Parser	76.32	60.42	67.44
	Post-processor	76.40	76.56	76.48
All brackets	Parser	88.55	88.15	88.35
	Post-processor	88.49	88.56	88.53

Table 24
Performance comparison of suggestion baseline, parser, and the NP bracketer post-processing the parser's output on test data.

EVALUATING	MODEL	*P*	*R*	*F*
NML JJP	Suggestions	94.29	56.81	70.90
	Parser	80.06	63.70	70.95
	Post-processor	79.44	78.67	79.05
All brackets	Parser	88.30	87.80	88.05
	Post-processor	88.23	88.24	88.23

This result is 7.04% lower than the figure previously achieved for complex NPs, despite the fact that unambiguous NPs are now included in the data. There are a number of reasons for this. Firstly, the test NPs produced by the parser may be incorrect, whereas the model is trained on gold-standard NPs. Also, the brackets that we rejected for crossing NP boundaries would introduce a noticeable amount of noise, and mean that the evaluation might not be entirely accurate. Finally, the POS tags used in these experiments are no longer gold-standard, as they come from the parser's output.

6.5.1 Parsing Evaluation. Finally, we can now put the rebracketed NPs back into the parser output and re-evaluate. This requires the additional task of labeling the brackets. There are only two labels to distinguish between (NML and JJP), and they can be inferred directly from the POS tag of the head. If it is a verb or an adjective, we label the node as JJP, and otherwise it is a NML. A small number of errors (a 0.42% drop in matched bracket F-score) are introduced by this method, because of annotation errors in the Penn Treebank POS tags and in our own annotation, as well as errors in head finding.

Tables 23 and 24 show the final results. A suggestion baseline is not shown for all brackets because they only apply to NMLs and JJPs and it is difficult to post-process the parser's output with them. The post-processor outperformed the parser by 9.04% and 8.10% on the development and test data, respectively. The post-processor has also improved on the suggestion baseline established earlier.

We measure statistical significance using a computer-intensive, randomized, stratified shuffling technique (Noreen 1989; Cohen 1995, §5.3) as implemented by Bikel.[7]

[7] http://www.cis.upenn.edu/∼dbikel/software.html.

The difference in all-brackets F-scores reported in Table 24 is statistically significant ($p \leq 0.0001$) using this test. Our results demonstrate the effectiveness of large-scale NP bracketing techniques, and show that internal NP structure can be recovered with better preformance than has ever been possible in the past.

7. Future Work

This work is the first to create and make use of a large-scale corpus of NP annotations. Our experiments with this new data have set a high benchmark for NP parsing. In many cases, there has been no previous work or state-of-the-art result to compare to, only experiments on a data set that is limited in scale and coverage. Our NP Bracketing experiments in Section 6, for example, set a new bar for what can be achieved, and demonstrate the applicability of supervised methods. There are now many directions for future work on the subject of NP parsing.

7.1 NP Annotation

In Section 3, we extended the Penn Treebank with NP annotations. For the first time, this widely used corpus can be used to train parsers to recover NP structure. We are aware of only one other corpus that has been annotated with a large volume of NP structure: the Biomedical Information Extraction Project (Kulick et al. 2004). Because these are the first NP annotation schemes, it seems probable that they can be improved.

The first category of NPs are those with genuinely flat structure, which are currently treated as implicitly right branching. For example, *John A. Smith* should be interpreted as a single unit, rather than as having left or right-branching structure. McInnes, Pedersen, and Pakhomov (2007) recognize monolithic NPs in their annotation of medical terms.

The second additional category is semantically indeterminate NPs. These NPs can be thought of as both left- and right-branching (i.e., a dependency should exist between all word pairs). Lauer (1995) found that 35 out of the 279 non-error NPs in his data set fit this category, for example, *city sewerage systems* and *government policy decisions*. It is the *government policy* in question in the latter example, but also *policy decisions* and *government decisions*, resulting in all three possible dependencies.

Marcus, Santorini, and Marcinkiewicz (1993) make some mention of indeterminate NPs, calling them *permanent predictable ambiguities*, a term they ascribe to Martin Kay. The example *a boatload of warriors blown ashore* is given, which is similar to the prepositional phrase attachment ambiguities in Hindle and Rooth (1993). The meanings of both attachments are true in cases like this: the *boatload* was *blown ashore*, and so were the *warriors*. Marcus et al. (1994) describe the *PPA* trace used in the Penn Treebank, which is applied to these permanent predictable ambiguities, or as we have called them, indeterminates. However *PPA* is also applied to cases of general ambiguity (those described in the following paragraphs), whereas we would separate the two.

The final category that we suggest is for ambiguous NPs. These NPs do have a left- or right-branching structure, although the annotator has no hope of determining which is correct. This may be because of technical jargon (e.g., *senior subordinated debentures*), or simply an ambiguity that cannot be resolved by the given context, as in the often cited PP-attachment example: *I saw the man with the telescope*. In these cases, there is a definite correct answer. The man either has a telescope, or a telescope is being used to do the seeing, but not both.[8] This differentiates these ambiguous cases from indeterminate NPs, where *both* readings are true.

None of the divisions just described are reflected in our corpus, and as a result, may have affected our experiments and/or their evaluations. For example, an NP bracketer training on a genuinely flat NP will learn that there is a dependency between the rightmost words when there is not. Similarly, an indeterminate NP (i.e., left flat by the annotator) suggests that the left-branching dependency is incorrect, when in fact both the left- and right-branching dependencies are correct. Adding these distinctions to the corpus may improve the performance of a bracketing model. However, we expect that any change would be small. This is because almost all NPs can be confidently assigned to left- or right-branching classes. We can present no gold-standard figures from the data, as the annotation of these additional NP structure categories has not been performed, but we can note that the annotator only marked 915 of the 60,959 inspected NPs as difficult (1.50%), and that in our experience, most of the difficulty comes from financial jargon, rather than the linguistic complications described here.

7.2 Parsing NP Structure

Our experiments in Section 5 highlighted the difficulty of parsing NPs. Many of the errors that occurred were due to a lack of lexical information and the productivity of nouns and noun phrases. One potential approach to solving these problems would be to incorporate the information sources that were successfully applied to NP Bracketing into a Collins-style parser. In particular, the possibility exists to include NER and Web 1T features as additional probability distributions in the model. Other parsers without a specialized NP submodel may make this process easier and/or more effective.

Our NP Bracketing work could be improved by finding a more effective process than the Barker (1998) algorithm. This is only one way to bracket complex NPs, which is a standard structured search problem. Another potential framing would be to treat the task as a sequence tagging problem, where the goal is to generate some number of brackets between individual tokens. The systems in Daumé III and Marcu (2004) and Bergsma and Wang (2007) function in a similar manner.

8. Conclusion

The addition of consistent, gold-standard, noun phrase structure to a large corpus is a significant achievement. We have shown that these annotations can be created in a feasible time frame with high inter-annotator agreement of 98.52% (measuring exact NP matches). In doing so, we have created a significantly larger corpus for analyzing NP structure than has ever been made available before. This is integrated with perhaps the most influential corpus in NLP. The large number of systems trained on Penn Treebank data can all benefit from the extended resource we have created.

In Section 5, we put the NP augmented Penn Treebank to use in training and evaluating the Collins (2003) parsing model. The results of these experiments demonstrated the difficulty that statistical methods have in bracketing NPs. The parsing model could not effectively adapt to the productivity of NP structure, and as a result, its performance

[8] In theory, the telescope could be with the man *and* used to do the seeing, but we will ignore this rather pathological possibility.

was lower than the baseline we set using deterministic rules. This baseline from the annotation tool's suggestion feature outperformed the parser by 5.81%.

Despite this, our analysis of Collins's model highlighted a number of interesting points. In particular, the continued importance of the base-NP submodel was a surprising result, as performance dropped spectacularly when it was removed. Our comprehensive error analysis showed that the largest cause of errors was a lack of lexical information in the training data.

Attempting to solve this problem, Section 6 saw the development of our NP Bracketing system. This is the first NP Bracketer that uses a supervised model to good effect and that can analyze NPs of arbitrary length and complexity. The initial simple NP bracketing experiments demonstrated that we could achieve performance on Lauer's small data set akin to that of previous researchers (e.g., Lauer 1995; Nakov and Hearst 2005). Our much larger data set from the Penn Treebank allowed us to build supervised models with even higher performance, however.

We moved onto the more realistic task of bracketing complex NPs. Utilizing the supervised model we built for simple NPs, and including a wide range of features, both novel and based on those used by other researchers, we achieved an excellent performance figure of 89.14% matched bracket F-score. These results demonstrated that complex NP Bracketing is an interesting task with much room for innovation.

Using this complex NP Bracketer, we constructed a post-processor for the parsing experiments from Section 5. In doing so, we finally outperformed the suggestion baseline and improved on the parser's result by 9.04% F-score. Our NP Bracketing system performs better than a state-of-the-art parsing model.

A widely used Collins-style parser, together with our NP post-processor, and trained using the extended corpus we created, is now able to identify sub-NP brackets with a level of accuracy that can be exploited by many practical applications. As a result, we have made it possible to increase performance on question answering, anaphora resolution, and many other downstream NLP tasks.

Appendix A: Annotation Guidelines

This document describes guidelines for bracketing NP structure in the Penn Treebank. These guidelines are in addition to the Treebank II Guidelines (Bies et al. 1995). They are also based on, and overlap with, the Addendum for BioMedical Annotation (Warner et al. 2004). An earlier version (0.9) of these guidelines was used in the annotation described in Vadas and Curran (2007), whereas this version was used in a subsequent pass over the data.

A.1. Bracketing NPs

The goal of our annotation is to identify and bracket multi-token premodifiers in NPs. Quirk et al. (1985, page 1321) describe such premodifiers, which include adjectives, participles, nouns, genitives, and adverbs. All of these items are modifiable themselves, and this is precisely the behavior that we have annotated. Indeed, NPs with multiple premodifiers can be recursive to an arbitrary depth (though more than three or four levels is unusual), and the underlying structure is by no means always right-branching. However, we can still resolve this ambiguity, as (with our emphasis)

> obscurity in premodification exists **only** for the hearer or reader who is unfamiliar with the subject concerned ... (Quirk et al. 1985, page 1343)

Thus, our most difficult cases come from the financial jargon of the *Wall Street Journal*, but the correct bracketing of most NPs is simple to ascertain.

The main change described in these guidelines is a different way of representing NP structure. Treebank II Style is to leave NPs flat, not specifying additional structure. In our extension, we assume a right-branching structure in all NPs, and mark explicitly any left-branching constituents. As most NPs are right-branching, this reduces the amount of bracketing required and thus increases legibility. This means that NPs like this one do not need further bracketing:

```
(NP (DT The) (JJ average)
  (JJ seven-day) (NN compound) (NN yield) )
```

And the implicit structure represented is:

```
(NP (DT The)
  (NODE (JJ average)
    (NODE (JJ seven-day)
      (NODE (NN compound)
        (NODE (NN yield) ) ) ) ) )
```

When a left-branching modifier is present, as in this NP,

```
(NP (NN lung) (NN cancer) (NNS deaths) )
```

it is bracketed explicitly. To specify that *lung cancer* is a constituent, we insert a bracket around those words:

```
(NP
  (NML (NN lung) (NN cancer) )
  (NNS deaths) )
```

Though less frequent, brackets can also be necessary in non-base-NPs, as in these examples:

```
(NP-SBJ
  (NML (JJ former)
    (NAC (NNP Ambassador)
      (PP (TO to)
        (NP (NNP Costa) (NNP Rica) ) ) ) )
  (NNP Francis) (NNP J.) (NNP McNeil) )

(NP
  (NML
    (NP (NN Wendy) (POS ’s) )
    (NNP International) )
  (NNP Inc.) )
```

In the first example, we join *former* and the `NAC` node, as he is formerly the Ambassador, not formerly Mr. McNeil.

Multiple words can be included in a bracket, and internal to the bracket are still implicitly right-branching.

```
(NP
  (NML (JJ chief) (JJ financial) (NN officer) )
  (NNP John) (NNP Pope) )

(NP
  (NML (JJ hot-dipped) (JJ galvanized) (NN sheet) )
  (NNS products) )
```

So the sheet is hot-dipped and galvanized, and the products are made of this sheet. Alternate, incorrect bracketings could suggest the galvanization is hot-dipped (a NML node around those two words) or that the products themselves are hot-dipped and galvanized (if no NML node was used).

New brackets can be nested, and this is needed quite often.

```
(NP
  (NML
    (NML (NNP New) (NNP York) )
    (NNP Stock) (NNP Exchange) )
  (JJ composite) (NN trading) )
```

This correct bracketing describes composite trading on the Stock Exchange of New York.

Note that we never alter existing Treebank brackets or POS tags, we only add new brackets to specify our extended representation. Similarly, we have not corrected errors that have been noticed during the annotation process. This is so that the corpus remains as comparable as possible to the original version. Pre-existing errors can mean that the correct extended annotation cannot possibly be implemented, however. In these cases, we try to mark up any constituents that we still can, while not adding any brackets that are incorrect. This often results in the opposite to what is done in the normal case.

```
(NP
  (NP (DT the) (NNP Carper) (POS ’s) )
  (NNP Creek)
  (NN wine) )
```

In this example, the determiner should be outside the inner NP, so that it has scope over *wine*. Normally, we would bracket *the Carper* to separate it from the possessive (see Section A.2.7), but that is incorrect here. Similarly, we do not bracket *the Carper's Creek* because it would include *the*. This would be incorrect, as *the* is the determiner for the overall NP, not just *Carper's Creek*.

A.1.1 Node Labels

We use two new node labels: NML and JJP. We have distinguished these from the existing NP and ADJP labels, so that we can analyze them separately. This approach has the advantage that they can be mapped back to the existing labels if needed. NML is used when the modifier's head is a noun, as in previous examples, whereas JJP is used when the head is adjectival, as in this example:

```
(NP (JJP (JJ dark) (JJ red) )
  (NN car) )
```

The label should also be JJP in cases where the head is a gerund.

```
(NP (DT the)
  (JJP (JJS fastest) (VBG developing) )
  (NNS trends) )
```

A JJP node is needed when an adverb modifies an adjective:

```
(NP
  (JJP (RB relatively) (JJR higher) )
  (NNS rates) )
```

Finally, we also apply the JJP label to coordinated adjectives premodifying a noun. In cases like these with multiple heads, only one head needs to be adjectival for the label to be JJP. We do not have a label similar to UCP.

```
(NP (PRP$ its)
  (JJP (JJ current)
    (CC and) (JJ former) )
  (NNS ratepayers) )
```

```
(NP (DT the)
  (JJP (JJ British)
    (CC and) (NNP U.S.) )
  (NNS troops) )
```

In all other cases (the vast majority), a NML label should be used. This means that cases with unusual heads, like the following one where DTs are being coordinated, are labeled NML.

```
(NP
  (NML (DT any)
    (CC or) (DT all) )
  (NNS warrants) )
```

If any POS tag has been incorrectly annotated, then the label used should reflect the correct POS tag, rather than propagate the error.

A.1.2 Ambiguous Cases

In general, if an annotator is unsure as to whether bracketing is needed, or if both alternatives seem equally likely, then they should leave the NP flat. The following NPs are examples of such semantically ambiguous cases. In the first, both dependencies are true (i.e., the players are in college, and they play basketball). The third example has a genuinely flat structure.

```
(NP (NN college) (NN basketball) (NNS players) )
```

```
(NP (NN army) (NN ordnance) (NN depot) )
```

```
(NP (NNP John) (NNP A.) (NNP Smith) )
```

A.1.3 Head Derivation

Head-finding rules for NML and JJP constituents are the same as for NP and ADJP nodes, respectively. For a detailed description of these rules, see Collins (1999, page 238).

In most cases, the head is either the right-most noun, or inside the right-most NML node.

This is more complicated with coordinated and apposited structures, which will have multiple heads. The individual heads can still be determined with the standard rules.

A.1.4 Identifying the Correct Bracketing

The bracketing task involves deciding which words belong together as constituents. It is often useful to reword the sentence to see whether a constituent makes sense. In doing so, the aim is to determine the dependencies that will be formed, that is, to create a syntactic structure which yields the correct semantic structure. Here are a few ways this can be done:

1. Inversion – In the following NP, we are deciding whether or not to bracket *other two*.

   ```
   (NP-LGS (DT the) (JJ other)
     (CD two) (JJ outside) (NNS bidders) )
   ```

   If we invert these words to *two other*, then the NP retains the same meaning. Therefore *other* does not modify *two* and they should not be bracketed.

2. Removal – This test involves trying to force one word to modify another by placing them side by side, removing the intervening text.
   In the example below, does *Japanese* modify *auto maker* or *Mazda Motor Corp*? If we remove *auto maker*, then the NP would not make sense, and so it must be the former. We have inserted the appropriate NML node.

   ```
   (NP (NML (JJ Japanese) (NN auto) (NN maker) )
     (NML (NNP Mazda) (NNP Motor) )
     (NNP Corp) )
   ```

3. Postmodifier – If we move a premodifier to the end of the NP, making it postmodify the head, then the correct bracketing should become clearer. In the following description of a car that is a certain shade of red,

   ```
   (NP (JJ tomato) (JJ red) (NN car) )
   ```

   if we change the NP to *red car that is tomato* then we get a meaning that doesn't make sense. As this is not the case, we know that *tomato* and *red* should be joined in a constituent.

   ```
   (NP
     (JJP (JJ tomato) (JJ red) )
     (NN car) )
   ```

A.2. Specific Cases

A.2.1 Coordinations

Coordinations are one of the most difficult structures to bracket in NPs. This is because of the multi-headed nature of such constructs. We should not read the next example as implicitly right-branching, but with dependencies between *Bill* and *and*, and *Ted* and *and*. It does not need further bracketing.

```
(NP (NNP Bill) (CC and) (NNP Ted) )
```

On the other hand, the following example does need the NML bracket shown:

```
(NP (DT the)
  (NML (NNPS Securities)
    (CC and) (NNP Exchange) )
  (NNP Commission) )
```

Otherwise, its implicit structure would be as follows:

```
(NP (DT the)
  (NODE
    (NODE (NNPS Securities) )
    (CC and)
    (NODE (NNP Exchange)
      (NODE (NNP Commission) ) ) ) )
```

The erroneous meaning here is *the Securities* and *the Exchange Commission*, rather than the correct *the Securities Commission* and *the Exchange Commission*.

Bracketing is also needed in the first of the following, or else the interpretation will be *rock stars* and *rock royalty*, which is clearly incorrect. However, this *is* the case in the second example (both the words and actions are rude) and so no new brackets are needed there.

```
(NP (NML (NN rock) (NNS stars) )
  (CC and)
  (NML (NN royalty) ) )
```

```
(NP (JJ rude) (NNS words)
  (CC and) (NNS actions) )
```

Also note that *royalty* is bracketed as a single word. This is because whenever one coordinated constituent is bracketed, all other constituents of the coordinate must be bracketed as well, even single tokens as seen here. This has changed since version 0.9 of these guidelines.

The implicit structure of the following NP is correct, as *rock stars* is already right-most.

```
(NP (NN royalty)
  (CC and) (NN rock) (NNS stars) )
```

However, this NP should be treated in the same way as the previous one. We therefore insert brackets around *rock stars* and *royalty* as before.

```
(NP (NML (NN royalty) )
  (CC and)
  (NML (NN rock) (NNS stars) ) )
```

If *any* constituent to be coordinated is multi-token (even right-most and implicitly correct ones), then *all* constituents of the coordinator must be explicitly bracketed. This is another change since the version 0.9 guidelines, which would not add any new brackets to this example.

Lists do *not* need any bracketing.

```
(NP (NNS cars)
  (, ,)
  (NNS trucks)
  (CC and) (NNS buses) )
```

This is true even when the conjunction is missing:

```
(NP
  (NP (DT no) (NN crack) (NNS dealers) )
  (, ,)
  (NP
    (NP (DT no) (JJ dead-eyed) (NNS men) )
    (VP (VBG selling)
      (NP
        (NP (JJ four-year-old) (NNS copies) )
        (PP (IN of)
          (NP (NNP Cosmopolitan) )))))
  (, ,)
  (NP
    (NP (DT no) (PRP one) )
    (VP (VBD curled)
      (PRT (RP up) )
      (PP-LOC (IN in)
        (NP (DT a) (NN cardboard) (NN box) )))))
```

However, the entire list may still need to be bracketed before being joined to words outside the list, as shown:

```
(NP
  (NP (NNP Mazda) (POS ’s) )
  (NNP U.S.)
  (NML (NNS sales)
    (, ,)
    (NN service)
    (, ,)
    (NNS parts)
    (CC and) (NN marketing) )
  (NNS operations) )
```

A list of attributes separated by commas does *not* need any bracketing:

```
(NP
  (JJ tricky)
  (, ,)
  (JJ unproven) (NN chip) (NN technology) )
```

This is because *tricky* and *unproven* are *not* being coordinated here. They are simply both acting as modifiers on *technology*, like in the NP: *big red car*.

Conjunctions between a *neither/nor* pair do *not* need any bracketing.

```
(NP-SBJ (DT Neither)
  (NP (NNP Lorillard) )
  (CC nor)
  (NP
    (NP (DT the) (NNS researchers) )
    (SBAR
      (WHNP-3 (WP who) )
      (S
        (NP-SBJ (-NONE- *T*-3) )
        (VP (VBD studied)
          (NP (DT the) (NNS workers) ))))))
```

A.2.2 Speech Marks

Tokens surrounded by speech marks should be bracketed:

```
(NP-PRD (DT a)
  (NML (‘‘ ‘‘) (JJ long) (NN term) (’’ ’’) )
  (NN decision) )
```

This includes when there is only a single token inside the speech marks, and when the speech marks are right-most:

```
(NP-PRD (DT a)
  (JJP (‘‘ ‘‘) (JJ long) (’’ ’’) )
  (NN decision) )

(NP-PRD (DT a)
  (NML (‘‘ ‘‘) (JJ long) (NN term) (’’ ’’) ) )
```

Note that the label of the bracket should reflect the internal head, as in the first example in the previous block, where JJP is used.

If the speech marks and the tokens they surround are the only items under the NP, then a new bracket should *not* be added.

```
(NP-PRD (‘‘ ‘‘) (JJ long) (NN term) (’’ ’’) )
```

The bracketing of speech marks has changed since the 0.9 version guidelines. Previously, the internal tokens were bracketed, whereas right-most speech marks were not.

Conventional editorial style for speech marks does not lend itself to bracketing easily. Because of this, there are a number of exceptions and corner cases when annotating NPs with speech marks. Firstly, in these examples:

```
(NP (‘‘ ‘‘)
  (NP-TTL (DT A) (NNP Place) (IN in) (NNP Time) )
  (, ,)
  (’’ ’’)
  (NP
    (NP (DT a) (JJ 36-minute) (JJ black-and-white) (NN film) )
    (PP (IN about)
      (NP
        (NP (DT a) (NN sketch) (NN artist) )
        (, ,)
        (NP
          (NP (DT a) (NN man) )
          (PP (IN of)
            (NP (DT the) (NNS streets) ) ) ) ) ) ) )
```

the comma serves to separate the film's title from its description, and the speech marks surround just the title. This causes a "crossing" constituent, as we cannot bracket the speech marks and the title together without including the comma. In these cases, we still add a NML bracket around the speech marks:

```
(NP
  (NML (`` ``)
    (NP-TTL (DT A) (NNP Place) (IN in) (NNP Time) )
    (, ,)
    ('' '') )
  (NP
    (NP (DT a) (JJ 36-minute) (JJ black-and-white) (NN film) )
    (PP (IN about)
      (NP
        (NP (DT a) (NN sketch) (NN artist) )
        (, ,)
        (NP
          (NP (DT a) (NN man) )
          (PP (IN of)
            (NP (DT the) (NNS streets) ) ) ) ) ) ) )
```

Many NPs contain a single opening or closing speech mark, whose partner is stranded in another constituent. For example, the following NP has only the opening speech mark:

```
(NP (DT the) (`` ``)
  (NML (NN type) (NN F) )
  (NN safety) (NN shape) )
```

In order to find the closing speech mark, we must look into the surrounding context:

```
(NP
  (NP (DT the) (`` ``)
    (NML (NN type) (NN F) )
    (NN safety) (NN shape) )
  (, ,)
  ('' '')
  (NP
    (NP (DT a) (JJ four-foot-high) (JJ concrete) (NN slab) )
    (PP (IN with)
      (NP (DT no) (NNS openings) ) ) ) )
```

In these cases, we could not bracket the speech marks properly without altering the existing structure. So once again, we do not add any new brackets in NPs such as this. In the next example, the speech marks have not been put in the right place:

```
(NP-PRD (`` ``) (DT a) (JJ worst-case) ('' '') (NN scenario) )
```

The determiner should be outside the speech marks. In cases such as these, the annotator should not follow the incorrect placement. Because no accurate bracketing can be inserted, no brackets should be added at all.

A.2.3 Brackets

These should be treated the same as speech marks, and bracketed as described in the previous section.

```
(NP (DT an)
  (JJP (-LRB- -LCB-) (VBG offending) (-RRB- -RCB-) )
  (NN country) )
```

An example of another corner case is shown here:

```
(NP (-LRB- -LCB-)
  (NML (NNP Fed) (NNP Chairman) )
  (NNP Alan)
  (-RRB- -RCB-)
  (NNP Greenspan) )
```

Once again, the tokens cannot be bracketed without a crossing constituent. We can still bracket *Fed Chairman*, but beyond that, no other brackets should be added.

A.2.4 Companies

Company names may need to be bracketed a number of ways. When there are postmodifiers such as Corp. or Ltd., the rest of the company needs to be separated if it is longer than one word.

```
(NP-SBJ
  (NML (NNP Pacific) (NNP First) (NNP Financial) )
  (NNP Corp.) )

(NP
  (NML (NNP W.R.) (NNP Grace) )
  (CC &) (NNP Co.) )

(NP
  (NML (NNP Goldman)
    (, ,)
    (NNP Sachs) )
  (CC &) (NNP Co.) )
```

Other identifiable nominal groups within the company name, such as locations, also need to be bracketed separately.

```
(NP
  (NP (NN today) (POS 's) )
  (NML (NNP New) (NNP England) )
  (NNP Journal) )
```

```
(NP (DT the)
  (NML (NNP Trade)
    (CC and) (NNP Industry) )
  (NNP Ministry) )
```

A.2.5 Final Adverbs

The tokens preceding a final adverb should be separated:

```
(NP (NML (NN college) (NNS radicals) )
  (RB everywhere) )
```

A.2.6 Names

Names are to be left unbracketed:

```
(NP (NNP Brooke) (NNP T.) (NNP Mossman) )
```

However, numbers, as well as *Jr.*, *Sr.*, and so forth, should be separated:

```
(NP
  (NML (NNP William) (NNP H.) (NNP Hudnut) )
  (NNP III) )
```

Titles that are longer than one word also need to be bracketed separately.

```
(NP
  (NML (NNP Vice) (NNP President) )
  (NNP John) (NNP Smith) )
```

A.2.7 Possessives

NPs preceding possessives need to be bracketed.

```
(NP (NML (NNP Grace) (NNP Energy) )
  (POS 's) )
```

A.2.8 Postmodifying Constituents

The words preceding a postmodificational constituent, such as a preposition or SBAR, do *not* need to be bracketed.

```
(NP
  (DT the) (JJ common) (NN kind)
  (PP (IN of)
    (NP (NN asbestos) )))
```

A.2.9 Unit Traces

This trace is necessary to make the unit (dollars in the following example) the head of the NP.

```
(NP (RB over) ($ $) (CD 27) (-NONE- *U*) )
```

If the NP is longer, and there are words to the right of the amount, then the trace should be inside the bracket.

```
(NP (DT a)
  (NML ($ $) (CD 27) (-NONE- *U*) )
  (NN charge) )
```

A.2.10 Unusual Punctuation

Sometimes a period indicating an acronym will be separated from the initial letter(s). In these cases, a bracket should be added to join them back together, as shown:

```
(NP (NNP Finmeccanica)
  (NML (NNP S.p) (. .) )
  (NNP A.) )
```

Some NPs also include final punctuation. These are mostly short fragmental sentences. In these cases, the rest of the NP should have a bracket placed around it:

```
(NP
  (NML
    (NML (NNP New) (NNP York) )
    (NNP City) )
  (: :) )
```

A.3. Future Improvements

Here we describe improvements to these guidelines and the bracketing scheme that we intend to carry out in the future. We noticed these issues during the first pass through the corpus, and all of them require another full pass.

A.3.1 Flat Structures

There are a number of NPs in the Penn Treebank that display genuinely flat structure. For some examples, refer back to Section A.1.2. We would like to distinguish these from the implicitly right-branching structures that make up the majority of the corpus. To do this, we intend to use a marker on the NP, NML, or JJP label itself, as shown:

```
(NP-FLAT (NNP John) (NNP A.) (NNP Smith) )

(NP
  (NML-FLAT (NNP John) (NNP A.) (NNP Smith) )
  (NNS apples) )
```

A.3.2 Appositions

Appositions are a multi-headed structure, similar but still different to coordination. They are extremely common throughout the Penn Treebank, and usually fit the pattern shown here, with a person's name and their position separated by a comma:

```
(NP-SBJ
  (NP (NNP Rudolph) (NNP Agnew) )
  (, ,)
  (NP
    (NP (JJ former) (NN chairman) )
    (PP (IN of)
      (NP (NNP Gold) (NNP Fields) (NNP PLC) ) ) ) )
```

We would like to mark these structures explicitly, so that they can be treated appropriately. This raises issues of what is and isn't an apposition (whether they are truly co-referential), and whether to discriminate between different types.

A.3.3 Head Marking

For some NPs, Collins's standard head-finding rules do not work correctly. In this example, *IBM* is the head, but *Australia* would be found.

```
(NP (NNP IBM) (NNP Australia) )
```

Marking heads explicitly would require a much larger degree of work, as NPs of length two would be ambiguous. All other annotation described here only needs to look at NPs of length three or more.

References

Abney, Steven. 1987. *The English Noun Phrase in its Sentential Aspects*. Ph.D. thesis, MIT, Cambridge, MA.

Atterer, Michaela and Hinrich Schütze. 2007. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469–476.

Barker, Ken. 1998. A trainable bracketer for noun modifiers. In *Proceedings of the Twelfth Canadian Conference on Artificial Intelligence (LNAI 1418)*, pages 196–210, Vancouver.

Bergsma, Shane and Qin Iris Wang. 2007. Learning noun phrase query segmentation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 819–826, Prague.

Bies, Ann, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. Technical report. University of Pennsylvania, Philadelphia, PA.

Bikel, Daniel M. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Black, Ezra, Steven Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith Klavans, Mark Liberman, Mitch Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*, pages 306–311, San Mateo, CA.

Brants, Thorsten and Alex Franz. 2006. Web 1T 5-gram version 1. Technical report. LDC Catalog No.: LDC2006T13. Google Research, Mountain View, CA.

Briscoe, Ted and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 41–48, Sydney.

Buckeridge, Alan M. and Richard F. E. Sutcliffe. 2002. Using latent semantic indexing as a measure of conceptual association for noun compound disambiguation. In *Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science (AICS-02)*, pages 12–19, Limerick.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 598–603, Providence, RI.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-00)*, pages 132–139, Seattle, WA.

Chiang, David and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-02)*, pages 1–7, Taipei.

Clark, Stephen and James R. Curran. 2007. Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 248–255, Prague.

Cocke, John and Jacob T. Schwartz. 1970. *Programming Languages and Their Compilers: Preliminary Notes*. Courant Institute of Mathematical Sciences, New York University, New York, NY.

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

Collins, Michael. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL-96)*, pages 184–191, Santa Cruz, CA, USA, June 24–27.

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 (ACL-97)*, pages 16–23, Madrid.

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

Collins, Michael. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Daumé III, Hal. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name`, implementation available at `http://hal3.name/megam/`.

Daumé III, Hal and Daniel Marcu. 2004. NP bracketing by maximum entropy tagging and SVM reranking. In Dekang Lin and Dekai Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 254–261, Barcelona.

Fayyad, Usama M. and Keki B. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI–93)*, pages 1022–1029, Chambery.

Fellbaum, Christiane, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Garside, Roger, Geoffrey Leech, and Geoffrey Sampson, editors. 1987. *The Computational Analysis of English: A Corpus-Based Approach*. Longman, London, UK.

Girju, Roxana, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Journal of Computer Speech and Language - Special Issue on Multiword Expressions*, 19(4):313–330.

Goodman, Joshua. 1997. Probabilistic feature grammars. In *Proceedings of the 5th International Workshop on Parsing Technologies (IWPT-97)*, September 17–20, 1997, pages 89–100, Cambridge, MA.

Hindle, Donald. 1983. User manual for Fidditch. Technical Report 7590-142, Naval Research Laboratory, Washington, DC.

Hindle, Donald. 1989. Acquiring disambiguation rules from text. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL-89)*, pages 118–125, Vancouver.

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

Hockenmaier, Julia. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh, Edinburgh.

Johnson, Mark. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Kasami, Tadao. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, MA.

King, Tracy Holloway, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC700 dependency bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8, Budapest.

Klein, Dan and Christopher D. Manning. 2001. Parsing with treebank grammars: empirical bounds, theoretical models, and the structure of the Penn Treebank. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL-01)*, pages 338–345, Toulouse.

Koehn, Philipp. 2003. *Noun Phrase Translation*. Ph.D. thesis, University of Southern California, Los Angeles, CA.

Kübler, Sandra. 2005. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *Proceedings of the Recent Advances in Natural Language Processing Conference (RANLP-05)*, September 21–23, 2005, pages 293–300, Borovets.

Kulick, Seth, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, and Lyle Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proceedings of BioLink Workshop at the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (BioLink-04)*, pages 61–68, Boston, MA.

Lapata, Mirella and Frank Keller. 2004. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of NLP tasks. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL-04)*, pages 121–128, Boston, MA.

Lauer, Mark. 1995. *Designing Statistical Language Learners: Experiments on Noun Compounds*. Ph.D. thesis, Macquarie University, Sydney.

Magerman, David. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Magerman, David. 1995. Statistical decision tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pages 276–283, Cambridge, MA.

Marcus, Mitchell. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, MA.

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. In *Proceedings of the Workshop on Human Language Technology (HLT-94)*, pages 114–119, Plainsboro, NJ.

Marcus, Mitchell, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

McInnes, Bridget, Ted Pedersen, and Serguei Pakhomov. 2007. Determining the syntactic structure of medical terms in clinical notes. In *Workshop on Biological, Translational, and Clinical Language Processing*, pages 9–16, Prague.

Melamed, I. Dan, Giorgio Satta, and Benjamin Wellington. 2004. Generalized multitext grammars. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 661–668, Barcelona.

Nakov, Preslav and Marti Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL-05)*, pages 17–24, Ann Arbor, MI.

Noreen, Eric W. 1989. *Computer Intensive Methods for Testing Hypotheses: An Introduction*. John Wiley & Sons, New York, NY.

Oepen, Stephan, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods Treebank: Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-02)*, pages 1253–1257, Taipei.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 311–318, Philadelphia, PA.

Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 433–440, Sydney.

Quirk, Randolph, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman, London.

Ramshaw, Lance A. and Mitchell Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94, Cambridge, MA.

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 (EMNLP-2)*, pages 1–10, Providence, RI.

Rehbein, Ines and Josef van Genabith. 2007. Treebank annotation schemes and parser evaluation for German. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 630–639, Prague.

Riezler, Stefan, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 271–278, Philadephia, PA.

Steedman, Mark. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Vadas, David and James R. Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-07)*, pages 240–247, Prague.

van Eynde, Frank. 2006. NP-internal agreement and the structure of the noun phrase. *Journal of Linguistics*, 42:139–186.

Wang, Wei, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 746–754, Prague.

Warner, Colin, Ann Bies, Christine Brisson, and Justin Mott. 2004. Addendum to the Penn Treebank II style bracketing guidelines: BioMedical Treebank annotation. Technical report, Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA.

Weischedel, Ralph and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical report. LDC Catalog No.: LDC2005T33, BBN Technologies, Cambridge, MA.

Younger, Daniel. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Zhang, Hao, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference - North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL)*, pages 256–263, New York, NY.