

# Large-Scale Supervised Models for Noun Phrase Bracketing

David Vadas and James R. Curran

School of Information Technologies

University of Sydney

NSW 2006, Australia

{dvadas1, james}@it.usyd.edu.au

## Abstract

Interpreting the structure of noun phrases (NPs) is important for many Natural Language Processing (NLP) tasks. This work extends the state-of-the-art in NP bracketing by: creating supervised models trained on a large annotated corpus; applying these to longer, more complex NPs; and using the resulting system to improve the output of the Bikel (2004) parser.

Using a large corpus of manually annotated Penn Treebank NPs we have developed a supervised model that brackets simple NPs with 93.01% F-score. We extend the evaluation to include longer, more complex NPs that are rarely dealt with in the literature, attaining 91.44% F-score. Finally, we implement a post-processing module that brackets NPs identified by the Bikel (2004) parser, which outperforms the parser itself by 8.13% F-score.

## 1 Introduction

Noun phrase (NP) bracketing is a requirement for the syntactic and semantic analysis of NPs. In the literature, e.g. Marcus (1980, p253) and Lauer (1995), the task is generally framed as follows: given a 3 word noun phrase like those below, decide whether it is left branching (1) or right branching (2).

((crude oil) prices) (1)

(world (oil prices)) (2)

NP bracketing is crucial for many Natural Language Processing (NLP) tasks. For example, question answering (QA) and anaphora resolution both require (potentially nested) candidate NPs, typically identified using a parser. If the answer or antecedent is not the complete NP, e.g. *crude oil* above, then it cannot be found.

Unfortunately, internal NP bracketing is not identified by parsers trained on the Penn Treebank, e.g. Collins (1999), because the Penn Treebank (Marcus et al., 1993), has flat NP structure as shown below:

(NP (NNP Air) (NNP Force) (NN contract))

Instead, researchers have attempted to solve the NP bracketing problem with unsupervised methods based on statistics from unannotated corpora (Lauer, 1995) or web hit counts (Lapata and Keller, 2004; Nakov and Hearst, 2005a).

The main contributions of this paper are large-scale supervised models trained on bracketed NPs from the entire Penn Treebank. The corpus also provides a large and representative test set. Using this data we significantly outperform previous approaches on the NP bracketing task. By incorporating a wide range of existing and novel features into the model, we increase performance by 8.87% in F-score over our best unsupervised system.

Further, we experiment with bracketing longer, *complex* NPs, by including NPs longer than three words, and which include non-nominal parts of speech (such as adjective, determiner, etc). These have been largely neglected in the literature, in contrast to *simple* NPs, i.e. those that are three words long and contain only nouns. We reimplemented the bracketing algorithm of Barker (1998) for complex noun phrases using the sophisticated supervised model we used for simple NPs. Our system achieves a performance of 91.44% F-score on matched brackets.

Finally, we apply these supervised models to the output of the Bikel (2004) parser. Our post-processor achieves an F-score of 77.27%, compared to parser output baseline of 69.14%. This is the first work to demonstrate that both simple and complex base-NPs can be bracketed with a high level of accuracy using supervised models.

## 2 Background

NP bracketing is similar to chunking (Ramshaw and Marcus, 1995), as both tasks aim to identify NP structure. Recursive NP bracketing, as in the CoNLL 1999 shared task and as performed by Daumé III and Marcu (2004) is closer still. However both these tasks are strictly less difficult than NP bracketing as defined in this paper, as they do not attempt to recover the full extent of sub-NP structure. This is in part because gold-standard annotations for this task have not been available in the past.

A basic method for solving the simple NP bracketing task was first described in Marcus (1980). This *adjacency* model compares the semantic association of words 1–2 to that between words 2–3. If the former is more likely, then the compound is left branching, otherwise it is right branching. Various methods of measuring the semantic association between a pair of words have been proposed for NP bracketing (Pustejovsky et al., 1993; Resnik, 1993) but they all depend on counting occurrences of bigrams in some corpus. Metrics such as  $\chi^2$  and mutual information can be used instead of the raw counts, and have been shown to perform well (Nakov and Hearst, 2005a).

Lauer (1995) proposes a new variation: the *dependency* model. In this case, we compare the semantic association of words 1–2 to that of words 1–3. This change is motivated by the dependencies that arise from the structure of the NP. We would expect a dependency between words 2–3 whether the compound was left or right branching, so there is no reason to analyse it.

Lauer (1995) demonstrated the superior performance of the dependency model using a test set of 244 (216 unique) noun compounds drawn from Grolier’s encyclopedia. This data has been used to evaluate most research since. Lauer uses Roget’s thesaurus to smooth words into semantic classes, and achieves 80.7% accuracy.

Lapata and Keller (2004) derive bigram probability estimates from web counts, and unlike Lauer, they use no smoothing beyond the lexical level, achieving 78.7% accuracy. Nakov and Hearst (2005a) 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) annotate a training set of 362 NPs drawn from WSJ text and use it to train a decision tree classifier, achieving 73.1% accuracy. Their feature set is made up of semantic information from WordNet. When they shuffled their data with Lauer’s to create a new test and training split, their accuracy increased to 83.1%, which may be a result of the  $\sim 10\%$  duplication in Lauer’s test set.

Barker (1998) describes an algorithm for bracketing a complex NP (described in Section 5) that reduces the problem to making a number of decisions on 3 word NPs. This system attains 62% and 65% accuracy on two different data sets for the simple NPs that are processed during the algorithm. Our supervised method performs as high as 96.19% on simple NPs, and so the performance for complex NPs in our implementation of Barker increases dramatically as well.

We have experimented with most of the features described in the literature: semantics; counts drawn from search engines and from a very large corpus of 1 trillion words; and additional lexical queries. We also implement novel features based on the NP’s context and the local POS and NER tags, and combine them all using a maximum entropy model.

## 3 Data

Vadas and Curran (2007) manually annotated all noun phrases in the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993). The annotation process involved inserting NML and/or JJP nodes to define each NP’s internal structure, as shown below:

```
(NP
  (NML (NNP Air) (NNP Force))
  (NN contract))
```

The agreement for the annotations was 98.52% (exact NP match) on WSJ Section 23 after discussion between the two annotators.

We now use the Vadas and Curran data to extract simple NPs from the Penn Treebank. If the last three children of an NP are nouns, then they became an example in our data set. Lauer (1995) used a similar approach of collecting all three noun sequences and then manually filtering them from Grolier’s encyclopedia.

If the first and second words are bracketed, then it is left branching, otherwise it is right branching. The annotation scheme does not currently

#	$H$	SEQUENCE	EXAMPLE
1053	0.13	( JJ JJ NNS )	big red cars
1353	0.60	( DT JJ NN NN )	the high interest rate
1480	0.55	( JJ NN NNS )	high interest rates
1761	1.00	( NNP NNP NNP )	John A. Smith
1793	1.00	(( NNP NNP ) NNP )	John Smith Co.

Table 1: Complex NP POS tag sequences

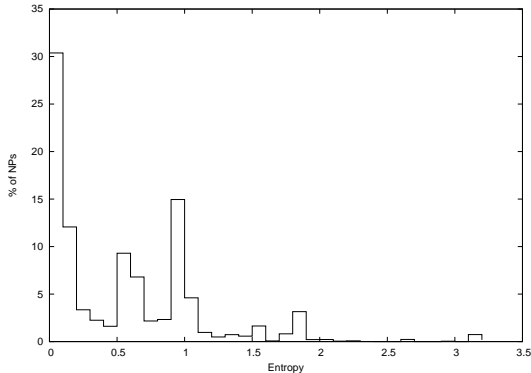


Figure 1: Entropy of NP POS tag sequences

mark flat base-NPs. Instead, all structure is either marked as left branching or implicitly right branching.

Note that because we are only looking at the right-most part of the NP, we know that we are not extracting any incomplete NPs. Finally, we remove examples where each word has the same NER tag, ignoring many of the flat base-NP cases such as *John A. Smith*.

This process results in 5582 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 sophisticated machine learning effectively, rather than using unsupervised methods.

We have also extracted another even larger data set of complex NPs, for the experiments in Section 5. For this set we retrieve an example for each base-NP of length three or more in the Penn Treebank. 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 leaves 36584 instances in our data set, which is two orders of magnitude larger than any that has been created previously.

Table 1 shows the most common POS tag sequences in our complex NP data set. The entropy  $H$  of the distribution of bracketings for the POS tag sequences gives an indication of the difficulty of the task. Larger entropy means that the sequence

COUNTS	MODEL	LAUER	PTB
Google	$\chi^2$ adjacency	72.95	79.86
	$\chi^2$ dependency	76.23	70.15
	Voters	83.61	83.98
MSN	$\chi^2$ adjacency	72.13	79.60
	$\chi^2$ dependency	74.59	69.76
	Voters	81.97	83.27
Web 1T	$\chi^2$ adjacency	74.59	80.40
	$\chi^2$ dependency	82.38	70.62
	Voters	83.20	81.60

Table 2: Unsupervised results for simple NPs

is more ambiguous, either because there are many bracketing alternatives to choose from or because the alternatives are close to equally likely.

Figure 1 shows a histogram of the entropy distribution across POS tag sequences. While 24% of all sequences have a single bracketing, the majority of sequences are ambiguous. There is a spike just below 1 bit, where sequences have two almost equally likely bracketings. This demonstrates that complex NP bracketing is far from a trivial task.

## 4 Experiments on Simple NPs

### 4.1 Unsupervised Approach

For these unsupervised experiments, we used all 5582 simple NPs as test data, and our hit counts come from search engine queries on the web, following Lapata and Keller (2004). We implement both the adjacency and dependency models, using the raw counts, the bigram probability  $P(w_i, w_j | w_j)$ , and the  $\chi^2$  measure. We also retrieved hit counts for a number of other variations that were proposed by Nakov and Hearst (2005a), such as the bigram joined by a hyphen to form a single token, or with a possessive marker.

The results using the best performing metric,  $\chi^2$ , are shown in Table 2. The voting result comes from choosing a subset of the web query variations, each of which votes for left or right branching. Following Nakov and Hearst (2005a) the results shown for these models are optimal for each test set, with no development set. When optimised on a held-out set, the results varied significantly, and were generally worse than choosing the best individual model. Our best result on Lauer’s data, 83.6%, is lower than Nakov and Hearst’s 89.3%. However, we haven’t implemented paraphrases or the morphological queries since they require a huge number of searches.

COUNT SOURCES	LAUER	PTB
Google / MSN	0.97	0.72
Google / Web 1T	0.92	0.60
MSN / Web 1T	0.93	0.81

Table 3: Correlation between counts

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. It is interesting to see how much the results vary, which indicates how sensitive this unsupervised method is to the counts it is based upon. Table 3 shows the correlation between the counts sources on the two data sets. This highlights the large degree of variability between our three sources of web counts especially on the larger Penn Treebank set. However, Nakov and Hearst (2005b) found that this variation did not have a significant impact on the performance of their NP bracketing system.

## 4.2 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 wide-coverage supervised NP bracketer.

We use the MegaM Maximum Entropy classifier (Daumé III, 2004) and discretise 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.

The initial features in our model were the counts used in the unsupervised experiments. For each query in Nakov and Hearst’s extended set, we get a left count, a right count, and a binary feature representing the left or right vote. We also included the corresponding probability and  $\chi^2$  metrics for the original queries (i.e. not including the extended Nakov and Hearst queries).

All of these features were defined over both the adjacency and dependency representations. The counts themselves come from Google and the Web 1T corpus, with separate features for each source. This first supervised model has 860 features in total, all based on the unsupervised counts.

MODEL	F-SCORE
Unsupervised, voting	84.14
All unsupervised features	90.59
All supervised features	90.59
– Web searches	91.94
– Web 1T corpus	93.01
– Lexical	92.20
– POS	92.56
– NER	92.20
– Context sentence	92.74
– Context window	92.65
– Semantic	92.74
All features	<b>93.01</b>

Table 4: Supervised results

The results on our Penn Treebank test set are shown in Table 4. The unsupervised voting system is included for comparison purposes. We can see that the supervised model with unsupervised features outperforms the unsupervised voting model by 6.45%. This improvement comes from the supervised model’s ability to weight the individual contributions of all of the unsupervised counts from Google and the Web 1T corpus.

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 have added lexical features for all unigrams, bigrams and the trigram within the NP. All of these features are labelled with the position of the n-gram within the NP.

Since we are bracketing NPs *in situ* rather than stand-alone NPs (like Lauer) we can exploit the context around the NP 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.

As we are using the Penn Treebank, we have access to gold-standard POS and NER tags (the latter from the BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein, 2005)). For every n-gram and context window feature, we also added generalised features by 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.

Finally, we incorporate semantic information from WordNet (Fellbaum, 1998). 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 additional feature types increase the number of features in the maximum entropy classifier to 88,568.

Table 4 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 supervised and unsupervised features each contribute equally, while removing the Web 1T counts does not decrease performance at all, probably because they overlap significantly with the web searches.

Of the supervised features, the lexical and NER are most important but all make a positive contribution. Our best performance of 93.01% F-score comes when using all features.

## 5 Experiments on Complex NPs

Up until now, all of our experiments (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 are made using the techniques described above for simple NPs. Barker’s algorithm is shown in Figure 2.

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

The complex NP data set is extracted from all of the Penn Treebank NPs annotated by Vadas and Curran (2007). We have extracted 36,584 complex NPs, which we split in a 9:1 ratio, giving 32,925 and 3659 training and test examples respectively.

### 5.1 Evaluation measures

The complex NP results are evaluated using several measures. Matching brackets is the standard Parseval evaluation method (Sekine and Collins, 1997). We also report implicit matching brackets which treats the structure as a binary tree, where implicit brackets are assumed to be right branch-

---

*w* is the current position of the window

1. *w* initially covers the last 3 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 2: Barker’s (1998) NP bracketing algorithm. Exact match measures the percentage of complex NPs that are entirely correct, and simple NPs measures the model’s performance on the 3 word NPs that are processed during Barker’s algorithm.

Matched brackets is a fairly tough evaluation, as the more easily identified flat NPs are not taken into account. For example, a baseline of always choosing right branching will score 0.0, as no explicit brackets are needed.

We only report F-score 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 simple NPs are different for each model, as the next three word NP to bracket depends on the decisions made previously for this complex NP, so the numbers are not directly comparable.

### 5.2 Complex NP Results

Our first experiment implemented Barker’s algorithm, using only the  $\chi^2$  dependency and adjacency methods. We only use counts from the Web 1T corpus, since performing web searches has become impractical with the increased data set size and NP length. The difficulty of complex NP bracketing can be seen by the drop in performance using these simple approaches, e.g. from 80.40% to 58.41% for the adjacency model.

We next applied our supervised approach to complex NPs. This is more complicated now as we need to extract a training set of three word win-

MODEL	MATCHED BRACKETS			IMPLICIT	EXACT	SIMPLE
	<i>P</i>	<i>R</i>	<i>F</i>			
Baseline – right branching	0.00	0.00	0.00	62.25	54.66	62.60
$\chi^2$ Dependency	23.47	52.26	32.40	40.91	32.66	52.53
$\chi^2$ Adjacency	24.47	43.72	31.37	49.34	35.86	58.41
All features	88.10	88.19	88.15	93.59	90.52	94.83
–Web 1T corpus	88.50	86.51	87.49	93.13	89.97	94.43
–Lexical	87.77	87.68	87.73	93.11	90.11	94.40
–POS	86.41	86.72	86.56	92.72	89.29	94.16
–NER	87.76	87.23	87.49	93.23	90.13	94.53
–Context sentence	89.58	88.85	89.22	93.98	91.25	95.24
–Context window	88.27	88.45	88.36	93.62	90.71	94.82
–Semantic	87.46	87.33	87.39	92.96	89.75	94.49
–Non-head words	85.34	85.90	85.62	92.58	89.04	94.11
–Border words	88.15	87.79	87.97	93.48	90.74	94.79
–POS tag sequence rule	88.33	87.79	88.06	93.48	90.41	94.73
Best (500 iterations)	92.41	90.48	91.44	95.16	93.03	96.19

Table 5: Complex NP results

dows from the complex NPs. To do this, we run Barker’s algorithm on the 32,925 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 produces 77,900 training examples.

We experiment with the features used for simple NPs as well as some novel features. Firstly, we added features encoding the non-head words when the window already contains a bracket, which increased performance by 2.5% on matched bracket F-score. Secondly, we add the bigram of the words on the NP border, that is, where it overlaps with the context. Lastly, we extracted POS tag sequences with low entropy, i.e. are quite unambiguous, and have a feature explicitly encoding their most common branching in the training data.

The results are shown in Table 5. The supervised methods significantly outperform the unsupervised methods, with a matched brackets F-score comparable to the Collins (1999) parser’s overall performance. We carried out a subtractive analysis of the feature types, finding that context sentence has a negative impact on performance, and our best result comes from removing it. We ran the final experiment using 500 rather than the default 100 iterations in MegaM, to allow the estimation to converge.

### 5.3 Parser Post-Processing

This final set of experiments use the complex NP models as a post-processing step for a parser.

EVALUATING	<i>P</i>	<i>R</i>	<i>F</i>
All brackets	88.63	88.29	88.46
NML JJP	77.38	62.49	69.14
Base-NP NML JJP	48.09	56.30	51.87

Table 6: Bikel (2004) parser performance

EVALUATING	<i>P</i>	<i>R</i>	<i>F</i>
All brackets	88.57	88.64	88.60
NML JJP	77.31	77.23	77.27
Base-NP NML JJP	88.78	78.97	83.59

Table 7: Post-processor performance

Since our data is simply bracketed structure for the Penn Treebank, we can train a parser model and evaluate it on NP structure. We train Bikel’s (2004) implementation of Collins’ (1999) parser on Sections 02–21 of the Penn Treebank, and test on all sentences in Section 00.

Table 6 displays the parser’s performance. It originally achieved 88.92% F-score on all constituents, but training with the NP bracketing structure makes the task more difficult, dropping the F-score to 88.46%. We are particularly interested in the performance on just the NML and JJP nodes, and on the base-NPs that we will bracket. These figures form the baseline that we aim to improve on with our post-processing module.

There are two factors that reduce the upper bound on post-processing performance. Firstly, the post-processor only brackets NPs identified by

MODEL	MATCHED BRACKETS			IMPLICIT	EXACT	SIMPLE
	<i>P</i>	<i>R</i>	<i>F</i>			
$\chi^2$ Dependency	15.47	51.71	23.81	40.32	36.47	46.24
$\chi^2$ Adjacency	16.21	47.66	24.19	46.82	39.03	51.49
All features	73.85	85.17	79.11	90.96	90.62	91.94
– Web 1T corpus	76.06	81.75	78.80	91.14	90.68	92.07
– Lexical	71.46	79.97	75.48	89.79	89.18	90.91
– POS	74.66	82.51	78.39	91.14	90.48	92.13
– NER	76.55	84.41	80.29	91.85	91.51	92.82
– Context sentence	76.11	84.79	80.22	91.69	91.31	92.63
– Context window	75.60	84.41	79.76	91.58	91.15	92.50
– Semantic	75.86	83.65	79.57	91.44	91.05	92.35
– Non-head words	72.96	82.76	77.55	91.40	90.35	92.40
– Border words	76.57	83.65	79.95	91.49	91.21	92.40
– POS tag sequence rule	76.18	83.90	79.86	91.62	91.18	92.58
– Parser	75.23	83.90	79.33	91.37	90.98	92.38
Best	77.70	83.90	80.68	92.06	91.45	92.95
Best (500 iterations)	79.40	84.54	81.89	92.75	92.35	93.66

Table 8: Complex NPs for parsing results

the Bikel parser, which are incorrect in approximately 10% of cases. Secondly, in the Vadas and Curran (2007) annotation scheme, NML and JJP nodes may appear in any NP, not just base-NPs.

Our method only applies to base-NPs, so we cannot achieve 100%, as NML and JJP nodes dominated by non-base-NPs cannot be corrected. The parser achieves 51.87% on the base-NP subset of nodes. If we were 100% correct in this subset, then the parser would still only achieve 89.06% F-score over all NML and JJP nodes.

We train the complex NP bracketer on gold-standard NPs from Sections 02–21, extracting 59,247 complex NPs that produce 99,432 three word training examples. The test 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 test set of 3005 complex NPs.

We then carry out a similar analysis as above for complex NPs. In these experiments, we introduce novel features based on the parser’s output. Firstly, the labels of the parent and grandparent of the NP, which have been informative in PCFG parsing, are included. Secondly, we add features for the head-word of the parent and the grandparent, and generalise these using their POS and NER tags. Finally, we add the Web 1T counts for the parent and grandparent head-words.

The results of these experiments, including subtractive analysis on the feature types, are shown in Table 8. Unfortunately, we find that many of the features are not helpful, and our best model utilises only the Web 1T, lexical, POS and non-head word features.

This is quite different to the results we encountered for complex NPs. We believe this is because the test NPs produced by the parser may be incorrect, while the model is trained on gold-standard NPs. Together with the brackets that we rejected for crossing NP boundaries, this would introduce a noticeable amount of noise, and we would expect lower performance in a number of feature types.

We now perform the additional task of labelling 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 are introduced by this method, because of annotation errors in the Penn Treebank POS tags and by Vadas and Curran (2007), as well as errors in head finding.

Having completed the bracketing, we can then reinsert the complex NP structure from the post-processing module back into the parser output. Table 7 shows the results. The unlabelled F-score on base-NPs was 84.35% (cf. 83.59% labelled), demonstrating the efficacy of our simple labelling process. The post-processing module improved on

the parser’s F-score (cf. Table 6) by 31.72% for base-NPs, and by 8.13% on all `NML` and `JJP` nodes. Finally, the post-processing system has increased performance on all constituents, removing some of the loss incurred in order to make NP bracketing possible.

## 6 Conclusion

In this paper, we have presented a comprehensive analysis of the noun phrase bracketing task, investigating the performance of many different data sets, features, and models. We started with compound noun phrases, as in Lauer (1995), and used models developed for that task to bracket longer, complex NPs with Barker’s (1998) algorithm. Finally, we incorporate the complex NP model into a post-processor for the Bikel (2004) parser.

In every evaluation we carried out, our supervised model improved significantly upon the baseline unsupervised results. Although this is not surprising, it is the first time that sufficient annotated data has been available for supervised methods to be competitive on this task. We achieved 93.01% F-score on our simple NP test set, compared to an unsupervised model’s 84.14%. With complex NPs, our improvement was even greater: 91.44% F-score on matched brackets, compared to 32.40%. Our supervised approach also incorporates several novel feature types extracted from within and surrounding the NP to be bracketed.

Vadas and Curran (2007) found that inter-annotator agreement for NP bracketing was 98.52% (exact NP match), so while our best result of 93.03% is competitive, there is still room for significant improvement. We intend to continue these experiments with more detailed feature analysis, and an exploration of different machine learning methods. There are also possible improvements to Barker’s (1998) algorithm, e.g. incorporating the probability with which each decision is made.

Using the large amount of Penn Treebank data annotated by Vadas and Curran (2007), we have shown that complex NP bracketing can be performed with a level of accuracy necessary to be exploited by many practical applications. Our system can be used in conjunction with a widely used parser in order to identify sub-NP structure, thereby increasing performance on question answering, anaphora resolution, and many other downstream NLP tasks.

## Acknowledgements

We would like to thank Preslav Nakov, for providing us with the code and data for his NP bracketing system; and the anonymous reviewers for their helpful feedback. This work has been supported by the Australian Research Council under Discovery Projects DP0453131 and DP0665973.

## References

- Ken Barker. 1998. A trainable bracketer for noun modifiers. In *Proceedings of the Twelfth Canadian Conference on Artificial Intelligence (LNAI 1418)*, pages 196–210. Vancouver.
- Dan Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.
- Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. Technical report.
- Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.
- Hal Daumé III. 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/>.
- Hal Daumé III and Daniel Marcu. 2004. Np bracketing by maximum entropy tagging and svm reranking. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 254–261. Association for Computational Linguistics, Barcelona, Spain.
- Usama M. Fayyad 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, France.
- Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA USA.
- Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel An-tohe. 2005. On the semantics of noun compounds. *Journal of Computer Speech and Language - Special Issue on Multiword Expressions*, 19(4):313–330.
- Mirella Lapata 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*, pages 121–128. Boston.
- Mark Lauer. 1995. Corpus statistics meet the compound noun: Some empirical results. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, MA.
- Mitchell Marcus. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, MA.
- Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- Preslav Nakov and Marti Hearst. 2005a. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of CoNLL-2005, Ninth Conference on Computational Natural Language Learning*. Ann Arbor, MI.

- Preslav Nakov and Marti Hearst. 2005b. A study of using search engine page hits as a proxy for n-gram frequencies. In *Proceedings of the Recent Advances in Natural Language Processing Conference*. Borovets, Bulgaria.
- James Pustejovsky, Sabine Berger, and Peter Anick. 1993. Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331–358.
- Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*. Cambridge MA, USA.
- Philip Resnik. 1993. *Selection and Information: A Class-Based Approach to Lexical Relationships*. Ph.D. thesis, University of Pennsylvania.
- Satoshi Sekine and Michael Collins. 1997. EVALB bracket scoring program. Available from <http://nlp.cs.nyu.edu/evalb/>.
- David Vadas and James R. Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*.
- Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical report.