

# Parsing Internal Noun Phrase Structure with Collins' Models

David Vadas and James R. Curran

School of Information Technologies

University of Sydney

NSW 2006, Australia

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

## Abstract

Collins' widely-used parsing models treat noun phrases (NPs) in a different manner to other constituents. We investigate these differences, using the recently released internal NP bracketing data (Vadas and Curran, 2007a). Altering the structure of the Treebank, as this data does, has a number of consequences, as parsers built using Collins' models assume that their training and test data will have structure similar to the Penn Treebank's.

Our results demonstrate that it is difficult for Collins' models to adapt to this new NP structure, and that parsers using these models make mistakes as a result. This emphasises how important treebank structure itself is, and the large amount of influence it can have.

## 1 Introduction

Collins' parsing models (Collins, 1999) are widely used in natural language processing (NLP), as they are robust and accurate for recovering syntactic structure. These models can be trained on a wide variety of domains and languages, such as biological text, Chinese and Arabic. However, Collins' models were originally built for the Penn Treebank (Marcus et al., 1993), and as such, are predisposed to parsing not only similar text, but also similar structure.

This paper deals with the effects of assuming such a structure, after the Treebank has been altered. We

focus on noun phrases (NPs) in particular, for two reasons. Firstly, there are a number of intricacies as part of Collins' model in this area. Indeed, a Collins-style parser uses a different model for generating NPs, compared to all other structures. Secondly, we can make use of our previous work in annotating internal NP structure (Vadas and Curran, 2007a), which gives us a ready source of Penn Treebank data with an altered structure.

Using this extended corpus, we make a number of alterations to the model itself, in an attempt to improve the parser's performance. We also examine the errors made, focusing on the altered data in particular, and suggest ways that performance can be improved in the future.

Next, we look at what effect the NP bracketing structure has. The existing Penn Treebank uses a generally flat structure (especially in NPs), but when internal NP brackets are introduced this is no longer the case. We determine what is the best representation to use for these new annotations, and also examine why that is.

Finally, we experiment with the task of identifying apposition within NPs, using manually annotated data as a gold standard. Although we find that this is a relatively easy task, the parser's performance is very low. The reasons for why Collins' models are unable to recover this structure are then explored.

The work described here raises questions about how researchers create NLP models, which may be for any task and not just parsing. Implicitly assuming that data will always retain the same structure can, as the results described here show, cause many problems for researchers in the future.

## 2 Background

We present here a brief introduction to Collins’ (1999) models, focusing in particular on how they generate NPs. All of the models use a Probabilistic Context Free Grammar (PCFG), with the Penn Treebank training data used to define the grammar, and to estimate the probabilities of the grammar rules being used. The CKY algorithm is then used to find the optimal tree for a sentence.

The grammar is also lexicalised, i.e. the rules are conditioned on the token and the POS tag of the head of the constituent being generated. Estimating probabilities over the grammar rule, the head and the head’s POS is problematic because of sparse data problems, and so generation probabilities are broken down into smaller steps. Thus, instead of calculating the probability of the entire rule, we note that each rule can be framed as follows:

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

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. If we make independence assumptions between the modifiers, then using the chain rule yields the following equations:

$$P_h(H|Parent, h) \quad (2)$$

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

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

So the head is generated first, then the left and right modifiers (conditioned only on the head<sup>1</sup>) afterwards.

Now this differs for base-NPs, which use a slightly different model. Instead of conditioning on the head, the current modifier is dependant on the previous modifier, resulting in a sort of bigram model. Formally, equations 3 and 4 above are changed as shown below:

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

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

<sup>1</sup>There are also distance measures, the subcategorisation frame, etc, but they are not relevant for this discussion as they do not affect NPs.

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. However, `prices` is the head that would be found. Using the special NP submodel thus results in the correct behaviour. 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.

The separate NP submodel also allows the parser to learn NP boundaries effectively, i.e. it is rare for words to precede `the` in an NP; and creates a distinct X-bar level, which Collins notes is helpful for the parser’s performance.

In order 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 relabelled as NPB. For consistency, an extra NP bracket is inserted around NPB nodes not already dominated by an NP. These NPB nodes are reverted before evaluation.

In our previous work (Vadas and Curran, 2007a), we annotated the full structure of NPs in the Penn Treebank. This means that the true head can be identified, which may remove the need to condition on the previous modifier. We will experiment with this in Section 3.2.

### 2.1 Treebank Structure

Other researchers have also looked at the effect of treebank structure upon parser performance. Collins himself notes that binary trees would be a poor choice, as the parser loses some context sensitivity, and the distance measures become ineffective. He advocates one level of bracketing structure per X-bar level.

Goodman (1997) explicitly converts trees to a binary branching format as a preprocessing step, in order to avoid these problems. Johnson (1998) finds that the performance of simple PCFGs can be improved through tree transformations, while Klein and Manning (2001) observe that some simple tree transforms can increase parser speed. The variation shown in these approaches, all to the same task, highlights the difficulty in identifying optimal tree structure.

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

Table 1: Parser performance

The issue of treebank structure extends to other languages as well, and implies further difficulties when comparing between languages. 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 the treebank structure also affects evaluation methods.

### 3 Internal NP Brackets

We begin by analysing the performance of Collins’ model, using the Vadas and Curran (2007a) data. This additional level of bracketing in the Penn Treebank consists of NML and JJP brackets to mark internal NP structure, as shown below:

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

In the first example, a NML bracket has been inserted around *crude oil* to indicate that the NP is left-branching. In the second example, we do not explicitly add a bracket around *oil prices*, but the NP should now be interpreted implicitly as right-branching.

The experiments are carried out using the Bikel (2004) implementation of the Collins (1999) parser. We use Sections 02-21 for training, and report labelled bracket precision, recall and F-scores for all sentences in Section 00.

Firstly, we compare the parser’s performance on the original Penn Treebank to when it is retrained with the new NML and JJP bracketed version. Table 1 shows that the new brackets make parsing marginally more difficult overall (by about 0.5% in F-score). However, if we evaluate only the original structure, by excluding the new NML and JJP brackets, then we find that the F-score has dropped by only a negligible amount. This means that the drop in overall performance results from low accuracy on the new NML and JJP brackets.

	PREC.	RECALL	F-SCORE
Overall	88.09	87.77	87.93
Original structure	87.92	88.68	88.30
NML and JJP brackets only	100.00	53.54	69.74

Table 2: Parser performance with relabelled brackets

Bikel’s parser does not come inbuilt with an expectation of NML or JJP nodes in the treebank, and so it is not surprising that these new labels cause problems. For example, head-finding for these constituents is undefined. Also, as we described previously, NPs are treated differently in Collins’ model, and so changing their structure could have unexpected consequences.

In an attempt to remove any complications introduced by the new labels, we relabelled NML and JJP brackets as NP and ADJP, and then retrained again. These are the labels that would be given if internal NP structure was originally bracketed with the rest of the Penn Treebank. This relabelling 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 eliminates the need for any change to the parser itself.

The results in Table 2 show that this is not the case, as the overall F-score has dropped another 0.5%. The NML and JJP brackets cannot be evaluated directly in this experiment, but we can compare against the corpus without relabelling, and count correct bracketings whenever a test NP matches a gold NML. The same is done for ADJP and JJP brackets. This results in a precision of 100%, because whenever a NML or JJP node is seen, it has already been matched against the gold-standard. Also, some incorrect NP or ADJP nodes are in fact false NML or JJP nodes, but this difference cannot be recovered.

We carried out a visual inspection of the errors that were made in this experiment, but 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 example NP below:

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

	PREC.	RECALL	F-SCORE
Overall	88.51	88.07	88.29
Original structure	88.78	88.86	88.82
NML and JJP brackets only	75.27	58.33	65.73

Table 3: Performance with correct head-finding

We conclude that the model was not able to generalise 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.

### 3.1 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 chooses the left-most child as the head. This is incorrect in most `NMLS`, where the head is usually the rightmost child.

We define `NML` and `JJP` rules in the parser data file, copying those used 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 retrained and ran the parser again with this change, and achieve the results in Table 3.

Once again, we are surprised to find that the F-score has been reduced, though only by 0.06% overall in this case. This drop comes chiefly from the `NML` and `JJP` brackets, whose performance has dropped by about 2%. As before, we scanned the errors in search of an explanation; however, there was no readily apparent pattern. It appears that conditioning on the incorrect head is simply helpful when parsing some sentences, and instances where the correct head gives a better result are less frequent.

### 3.2 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.

		PREC.	RECALL	F-SCORE
1	Overall	72.11	87.71	79.14
	Original structure	72.09	88.19	79.33
	NML /JJP brackets only	72.93	69.58	71.22
2	Overall	87.37	87.17	87.27
	Original structure	87.75	87.65	87.70
	NML /JJP brackets only	72.36	69.27	70.78
3	Overall	86.83	86.46	86.64
	Original structure	86.90	88.66	87.77
	NML /JJP brackets only	48.61	3.65	6.78

Table 4: Performance with the base-NP model off

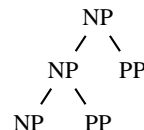


Figure 1: An unlikely structure

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, i.e. equations 3 and 4 are always used.
2. Removing the preprocessing step that creates `NPB` nodes. This alteration will have the same effect as the one above, 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 are in Table 4, and in all cases the overall F-score has decreased. In the 1st change, to `isBaseNP()`, performance on `NML` and `JJP` brackets has actually increased by 3.78% F-score, although the original structure is almost 10% worse. The 2nd 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 3rd 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,

ERROR	#	%	FP	FN	EXAMPLE	
Modifier attachment	213	38.04	56	157		
NML	122	21.79	21	101	lung cancer deaths	×
Internal 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 postmodifiers	9	1.61	1	8	(Kawasaki Heavy Industries) Ltd.	✓
Mislabelling	92	16.43	30	62	(ADJP more influential) role	✓
Conjunctions	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 and brackets	35	6.25	0	35	("closed-end")	✓
Clearly wrong bracketing	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	×
Conjunction	5	0.89	5	0	(NP a (NML savings and loan))	×
Structural	8	1.43	3	5	(NP ... construction spending) (VP (VBZ figures) ...)	×
Other	14	2.50	8	6		
Total	560	100.00	180	380		

Table 5: Error analysis

structures such as that in Figure 1, which *never* occur in the Treebank, are given too high a probability. The parser needs to know where NPs will not recurse anymore (when they are basal), so that it can generate the correct flat structure. Furthermore, the 3rd change effectively treats NP and NML nodes as equivalent, and we have already seen that this is not true.

### 3.3 Error Analysis

So far, all our changes have had negative results. We need to look at the errors being made by the parser, so that any problems that appear can be solved. Accordingly, we categorised every NML and JJP error through manual inspection. The results of this analysis are shown in Table 5, together with examples of the errors being made. Only relevant brackets and labels are shown in the examples, while the final column describes whether or not the particular bracketing shown is correct.

The most common bracketing error results in a modifier being attached to the wrong head. In the example, because there is no bracket around *lung cancer*, there is a dependency between *lung* and *deaths*, instead of *lung* and *cancer*. 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.

These errors occur because the ngrams that need to be bracketed simply do not exist in the training data. Looking for each of the 142 unique ngrams that were not bracketed, we find that 93 of them do

not occur in Sections 02-21 at all. A further 17 of the ngrams 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, an outside source of information must be consulted, as the lexical information is currently not available.

The next largest source of errors is mislabelling 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 preparation of the corpus that the labels of some NPs were inconsistent (Vadas and Curran, 2007a). The previous relabelling 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, which is not feasible here.

Conjunctions are another significant source of errors, and are quite a difficult problem. This is because coordinating multi-token constituents requires brackets around each of the constituents, as well as a further bracket around the entire conjunction. 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 weren't for the possessive. In

particular, `NML` nodes beginning with a determiner are 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’ model does not generate punctuation as it does other constituents. There is also less need for speech marks and brackets to be correct, as the standard evaluation does not find an error when they are placed in the wrong constituent. The justification for this is that during the annotation process, they were given the lowest priority, and are thus inconsistent.

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 is not bracketed explicitly) or because they dominate only a single token. Single token `NMLS` exist only in conjunctions, 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. This data does not need to be annotated, as we demonstrated in previous NP bracketing experiments (Vadas and Curran, 2007b), which attained a positive result. However, incorporating the data into Collins’ model is still difficult. Our previous work only implemented a post-processor that ignored the parser’s output. There is still room for improvement in this area.

## 4 Bracket Structure

We have now seen how a Collins-style parser performs on internal NP structure, but a question remains about the structure itself: is it optimal for the parser? It may be argued that a better representation 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

	PREC.	RECALL	F-SCORE
Overall	87.33	86.36	86.84
Original structure	87.96	88.06	88.01
<code>NML</code> and <code>JJP</code> brackets only	82.33	74.28	78.10

Table 6: Explicit right-branching structure

both cases. Explicit right-branching brackets could also remove some of the clearly wrong bracket errors in Table 5.

The current bracketing guidelines do not mark right-branching constituents, they are simply assumed implicitly to be there. We can automatically add them in however, and then examine what difference this change makes. We find, in Table 6, that overall performance drops by 1.51% F-score.

This was 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.

On the other hand, the increased number of brackets has had a deleterious effect on the original brackets. This result suggests that it is better to leave right-branching structure implicit.

## 5 Appositions

The results in this paper so far, have been attained using noun modifier data. This final set of experiments however, focuses upon a different kind of internal NP structure: appositions. We thus show that the effects of treebank structure are important for a wider range of constructions, and demonstrate once again the difficulty experienced by a model that must adapt to altered data.

Appositions are a very common linguistic construction in English. They have been used in areas such as Information Extraction (Sudo et al., 2003) and Question Answering systems (Moldovan et al., 2003), however there is little work on automatically identifying them. Researchers have typically used simple patterns for this task, although the accuracy

of this method has not been determined. We will compare how well Bikel’s parser performs.

As in our previous experiments, we use the Penn Treebank, as it contains numerous appositions, such as the (slightly edited) example shown below:

```
(NP-SBJ
  (NP (NNP Darrell) (NNP Phillips) )
  ( , )
  (NP (NN vice) (NN president) ))
```

Appositional structure is, of course, not annotated in the Penn Treebank, and so we manually added gold-standard apposition brackets to the corpus. This was actually done during the annotation process for the Vadas and Curran (2007a) data. For example, we add a new bracket labelled `APP` to the previous noun phrase:

```
(NP-SBJ
  (APP
    (NP (NNP Darrell) (NNP Phillips) )
    ( , )
    (NP (NN vice) (NN president) )))
```

We should note that we only look at *nonrestrictive* apposition, i.e. where NPs are separated by punctuation (usually a comma, but also a colon or dash). Cases of *close* apposition, as in the *vice president Darrel Phillips* have been shown to have a different interpretation (Lee, 1952) and also present more difficult cases for annotation.

There are 9,082 `APP` constituents in the corpus, out of the 60,959 NPs (14.90%) that were manually inspected during the annotation process. We measured inter-annotator agreement by comparing against a second annotator on Section 23. This resulted in an F-score of 90.41%, with precision of 84.93% and recall of 96.65%. The precision was notably low because the second annotator inserted `APP` nodes into NPs such as the one shown below:

```
(NP
  (APP
    (NP
      (QP ($ $) (CD 1.7) (CD million) )
      (-NONE- *U*) )
    ( , )
    (CC or)
    (NP
      (NP (CD 21) (NNS cents) )
      (NP-ADV (DT a) (NN share) ))))
```

and while these cases are appositive, the first annotator did not insert an `APP` node when a conjunction was present.

	PREC.	RECALL	F-SCORE
Overall	86.24	87.60	86.92
APP brackets only	69.79	66.37	68.04
All other brackets	86.43	87.86	87.14

Table 7: Parser results with appositions

## 5.1 Experiments

Our first experiment uses a pattern matching technique, simply identifying appositions by looking for a pair of NPs separated by a comma or colon. This rule was then expanded to include other similar constituent labels: `NML`, `UCP`, `ADJ`, `NNP`, and `APP` itself, after noticing that errors were occurring in these cases.

Evaluating this approach, using the entire Penn Treebank as a test set, we achieved an F-score of 95.76%, with precision 95.53% and recall 95.99%. This result is very good for such a simplistic approach, and could be improved further by adding some additional patterns, such as when an adverb appears between the comma and the second NP.

Having set this very high baseline, we once again used Bikel’s (2004) parser in an attempt to find an improvement. This experiment includes the `NML` and `JJP` brackets in the data, and the parser is in its original state, without any of the alterations we made earlier. The results are shown in Table 7.

The parser’s performance on `APP` brackets is almost 30% F-score below the pattern matching approach, and it has also dropped 1.21% counting only the non-`APP` constituents. The reason for this very low performance arises from the way that Collins’ models treats punctuation, i.e. all tokens with a POS tag of `.` or `:`. The Collins (1997) model does not generate punctuation at all, and later models still do not treat punctuation the same way as other tokens. Instead, punctuation is generated as a boolean flag on the following constituent. As a result, the parser cannot learn that a rule such as `APP → NP , NP` should have high probability, because this rule is never part of the grammar. For this reason, the parser is unable to replicate the performance of a simple rule.

## 6 Conclusion

The results of this paper emphasise the strong relationship between a statistical model and the structure of the data it uses. We have demonstrated this by changing two different constructions in the Penn Treebank: noun modifiers, and appositions.

The annotation structure of the Vadas and Curran (2007a) data has also been validated by these results, which is actually a complement for the BioMedical guidelines (Warner et al., 2004), on which ours were based. It is not necessary to explicitly bracket right-branching constituents, and furthermore, it is harmful to do so. In addition, separating the NML and NP labels is advantageous, although our results suggest that performance would increase if the original Treebank annotations were made more consistent with our internal NP data.

Our results also demonstrate the necessity of having a base-NP submodel as part of Collins' models. This specialised case, while seeming unnecessary with our new internal NP brackets, is still required to attain a high level of performance.

Instead, the error analysis in Section 3.3 shows us the true reason why the parser's performance on internal NP brackets is low. NML and JJP brackets are difficult because they require specific lexical information, i.e. the exact words must be in the training data. This is because POS tags, which are very important when making most parsing decisions, are uninformative here. For example, they do not help at all when trying to determine whether a sequence of 3 NNS is left or right-branching.

Our previous work in NP bracketing (Vadas and Curran, 2007b) is positive evidence that this is the correct direction, although incorporating such a submodel back into Collins' model in place of the existing NP submodel, would be a further improvement. It also remains to be seen whether the results observed here would apply to other parsing models.

This work has demonstrated the large effect that data structure can have on a standard NLP tool. It is important that any system, not just parsers, ensure that they perform adequately when faced with changing data. Otherwise, assumptions made today will cause problems for researchers in the future.

## Acknowledgements

We would like to thank Matthew Honnibal for his annotation of the apposition data; 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

- Dan Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.
- Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23.
- Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.
- Joshua Goodman. 1997. Probabilistic feature grammars. In *Proceedings of the International Workshop on Parsing Technologies*.
- Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.
- Dan Klein and Christopher D. Manning. 2001. Parsing with treebank grammars: empirical bounds, theoretical models, and the structure of the Penn Treebank. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 338–345. Toulouse, France.
- Sandra Kübler. 2005. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *Proceedings of RANLP 2005*. Borovets, Bulgaria.
- Donald W. Lee. 1952. Close apposition: An unresolved pattern. *American Speech*, 27(4):268–275.
- Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maierano. 2003. COGEX: A logic prover for question answering. In *Proceedings of HLT-NAACL 2003*, pages 166–172.
- Ines Rehbein 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.
- Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL-03)*, pages 224–231.
- David Vadas and James R. Curran. 2007a. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*.
- David Vadas and James R. Curran. 2007b. Large-scale supervised models for noun phrase bracketing. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING-2007)*. Melbourne, Australia.
- Colin Warner, Ann Bies, Christine Brisson, and Justin Mott. 2004. Addendum to the Penn Treebank II style bracketing guidelines: BioMedical Treebank annotation. Technical report.