# Parsing Noun Phrase Structure with CCG

**David Vadas** and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{dvadas1, james}@it.usyd.edu.au

## Abstract

Statistical parsing of noun phrase (NP) structure has been hampered by a lack of gold-standard data. This is a significant problem for CCGbank, where binary branching NP derivations are often incorrect, a result of the automatic conversion from the Penn Treebank.

We correct these errors in CCGbank using a gold-standard corpus of NP structure, resulting in a much more accurate corpus. We also implement novel NER features that generalise the lexical information needed to parse NPs and provide important semantic information. Finally, evaluating against DepBank demonstrates the effectiveness of our modified corpus and novel features, with an increase in parser performance of 1.51%.

## 1 Introduction

Internal noun phrase (NP) structure is not recovered by a number of widely-used parsers, e.g. Collins (2003). This is because their training data, the Penn Treebank (Marcus et al., 1993), does not fully annotate NP structure. The flat structure described by the Penn Treebank can be seen in this example:

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

CCGbank (Hockenmaier and Steedman, 2007) is the primary English corpus for Combinatory Categorial Grammar (CCG) (Steedman, 2000) and was created by a semi-automatic conversion from the Penn Treebank. However, CCG is a binary branching grammar, and as such, cannot leave NP structure underspecified. Instead, *all* NPs were made right-branching, as shown in this example:

```
(N
  (N/N lung)
  (N
    (N/N cancer) (N deaths) ) )
```

This structure is correct for most English NPs and is the best solution that doesn't require manual reannotation. However, the resulting derivations often contain errors. This can be seen in the previous example, where `lung cancer` should form a constituent, but does not.

The first contribution of this paper is to correct these CCGbank errors. We apply an automatic conversion process using the gold-standard NP data annotated by Vadas and Curran (2007a). Over a quarter of the sentences in CCGbank need to be altered, demonstrating the magnitude of the NP problem and how important it is that these errors are fixed.

We then run a number of parsing experiments using our new version of the CCGbank corpus. In particular, we implement new features using NER tags from the BBN Entity Type Corpus (Weischedel and Brunstein, 2005). These features are targeted at improving the recovery of NP structure, increasing parser performance by 0.64% F-score.

Finally, we evaluate against DepBank (King et al., 2003). This corpus annotates internal NP structure, and so is particularly relevant for the changes we have made to CCGbank. The CCG parser now recovers additional structure learnt from our NP corrected corpus, increasing performance by 0.92%. Applying the NER features results in a total increase of 1.51%.

This work allows parsers trained on CCGbank to model NP structure accurately, and then pass this crucial information on to downstream systems.



*Proceedings of ACL-08: HLT*, pages 335–343,
Columbus, Ohio, USA, June 2008. 

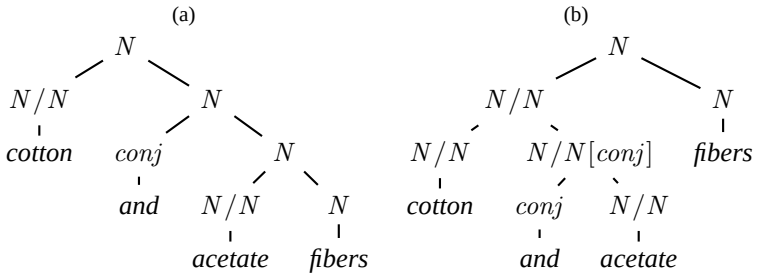



Figure 1: (a) Incorrect CCG derivation from Hockenmaier and Steedman (2007) (b) The correct derivation

## 2 Background

Parsing of NPs is typically framed as NP bracketing, where the task is limited to discriminating between left and right-branching NPs of three nouns only:

- *(crude oil) prices* – left-branching
- *world (oil prices)* – right-branching

Lauer (1995) presents two models to solve this problem: the adjacency model, which compares the association strength between words 1–2 to words 2–3; and the dependency model, which compares words 1–2 to words 1–3. Lauer (1995) experiments with a data set of 244 NPs, and finds that the dependency model is superior, achieving 80.7% accuracy.

Most NP bracketing research has used Lauer's data set. Because it is a very small corpus, most approaches have been unsupervised, measuring association strength with counts from a separate large corpus. Nakov and Hearst (2005) use search engine hit counts and extend the query set with typographical markers. This results in 89.3% accuracy.

Recently, Vadas and Curran (2007a) annotated internal NP structure for the entire Penn Treebank, providing a large gold-standard corpus for NP bracketing. Vadas and Curran (2007b) carry out supervised experiments using this data set of 36,584 NPs, outperforming the Collins (2003) parser.

The Vadas and Curran (2007a) annotation scheme inserts `NML` and `JJP` brackets to describe the correct NP structure, as shown below:

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

We use these brackets to determine new gold-standard CCG derivations in Section 3.

### 2.1 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) (Steedman, 2000) is a type-driven, lexicalised theory of grammar. Lexical categories (also called *supertags*) are made up of basic atoms such as $S$ (Sentence) and $NP$ (Noun Phrase), which can be combined to form complex categories. For example, a transitive verb such as `bought` (as in `IBM bought the company`) would have the category: $(S\backslash NP)/NP$. The slashes indicate the directionality of arguments, here two arguments are expected: an NP subject on the left; and an NP object on the right. Once these arguments are filled, a sentence is produced.

Categories are combined using combinatory rules such as forward and backward application:

$$X/Y \;\; Y \;\; \Rightarrow \;\; X \quad (>) \tag{1}$$

$$Y \;\; X\backslash Y \;\; \Rightarrow \;\; X \quad (<) \tag{2}$$

Other rules such as composition and type-raising are used to analyse some linguistic constructions, while retaining the canonical categories for each word. This is an advantage of CCG, allowing it to recover long-range dependencies without the need for post-processing, as is the case for many other parsers.

In Section 1, we described the incorrect NP structures in CCGbank, but a further problem that highlights the need to improve NP derivations is shown in Figure 1. When a conjunction occurs in an NP, a non-CCG rule is required in order to reach a parse:

$$conj \;\; N \;\; \Rightarrow \;\; N \tag{3}$$

This rule treats the conjunction in the same manner as a modifier, and results in the incorrect derivation shown in Figure 1(a). Our work creates the correct CCG derivation, shown in Figure 1(b), and removes the need for the grammar rule in (3).

Honnibal and Curran (2007) have also made changes to CCGbank, aimed at better differentiating between complements and adjuncts. PropBank (Palmer et al., 2005) is used as a gold-standard to inform these decisions, similar to the way that we use the Vadas and Curran (2007a) data.

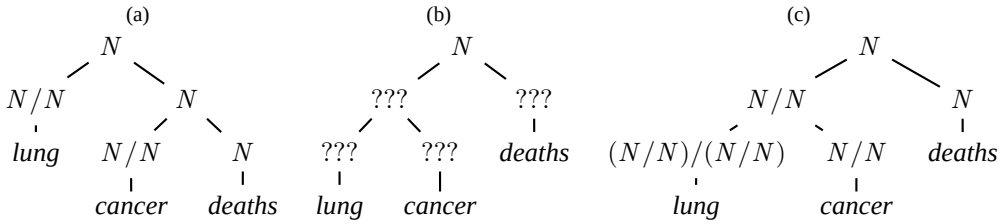



Figure 2: (a) Original right-branching CCGbank (b) Left-branching (c) Left-branching with new supertags

### 2.2 CCG parsing

The C&C CCG parser (Clark and Curran, 2007b) is used to perform our experiments, and to evaluate the effect of the changes to CCGbank. The parser uses a two-stage system, first employing a supertagger (Bangalore and Joshi, 1999) to propose lexical categories for each word, and then applying the CKY chart parsing algorithm. A log-linear model is used to identify the most probable derivation, which makes it possible to add the novel features we describe in Section 4, unlike a PCFG.

The C&C parser is evaluated on predicate-argument dependencies derived from CCGbank. These dependencies are represented as 5-tuples: $\langle h_f, f, s, h_a, l \rangle$, where $h_f$ is the head of the predicate; $f$ is the supertag of $h_f$; $s$ describes which argument of $f$ is being filled; $h_a$ is the head of the argument; and $l$ encodes whether the dependency is local or long-range. For example, the dependency encoding `company` as the object of `bought` (as in `IBM bought the company`) is represented by:

$$\langle bought, (S\backslash NP_1)/NP_2, 2, company, - \rangle \quad (4)$$

This is a local dependency, where `company` is filling the second argument slot, the object.

## 3 Conversion Process

This section describes the process of converting the Vadas and Curran (2007a) data to CCG derivations. The tokens dominated by `NML` and `JJP` brackets in the source data are formed into constituents in the corresponding CCGbank sentence. We generate the two forms of output that CCGbank contains: AUTO files, which represent the tree structure of each sentence; and PARG files, which list the word–word dependencies (Hockenmaier and Steedman, 2005).

We apply one preprocessing step on the Penn Treebank data, where if multiple tokens are enclosed by brackets, then a `NML` node is placed around those tokens. For example, we would insert the `NML` bracket shown below:

```
(NP (DT a) (-LRB- -LRB-)
  (NML (RB very) (JJ negative) )
  (-RRB- -RRB-) (NN reaction) )
```

This simple heuristic captures NP structure not explicitly annotated by Vadas and Curran (2007a).

The conversion algorithm applies the following steps for each `NML` or `JJP` bracket:

1. Identify the CCGbank *lowest spanning node*, the lowest constituent that covers all of the words in the `NML` or `JJP` bracket;
2. flatten the lowest spanning node, to remove the right-branching structure;
3. insert new left-branching structure;
4. identify heads;
5. assign supertags;
6. generate new dependencies.

As an example, we will follow the conversion process for the `NML` bracket below:

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

The corresponding lowest spanning node, which incorrectly has `cancer deaths` as a constituent, is shown in Figure 2(a). To flatten the node, we recursively remove brackets that partially overlap the `NML` bracket. Nodes that don't overlap at all are left intact. This process results in a list of nodes (which may or may not be leaves), which in our example is [`lung`, `cancer`, `deaths`]. We then insert the correct left-branching structure, shown in Figure 2(b). At this stage, the supertags are still incomplete.

Heads are then assigned using heuristics adapted from Hockenmaier and Steedman (2007). Since we are applying these to CCGbank NP structures rather than the Penn Treebank, the POS tag based heuristics are sufficient to determine heads accurately.

Finally, we assign supertags to the new structure. We want to make the minimal number of changes to the entire sentence derivation, and so the supertag of the dominating node is fixed. Categories are then propagated recursively down the tree. For a node with category $X$, its head child is also given the category $X$. The non-head child is always treated as an adjunct, and given the category $X/X$ or $X\backslash X$ as appropriate. Figure 2(c) shows the final result of this step for our example.

### 3.1 Dependency generation

The changes described so far have generated the new tree structure, but the last step is to generate new dependencies. We recursively traverse the tree, at each level creating a dependency between the heads of the left and right children. These dependencies are never long-range, and therefore easy to deal with. We may also need to change dependencies reaching from inside to outside the NP, if the head(s) of the NP have changed. In these cases we simply replace the old head(s) with the new one(s) in the relevant dependencies. The number of heads may change because we now analyse conjunctions correctly.

In our example, the original dependencies were:

$$\langle lung, N/N_1, 1, deaths, - \rangle \quad (5)$$

$$\langle cancer, N/N_1, 1, deaths, - \rangle \quad (6)$$

while after the conversion process, (5) becomes:

$$\langle lung, (N/N_1)/(N/N)_2, 2, cancer, - \rangle \quad (7)$$

To determine that the conversion process worked correctly, we manually inspected its output for unique tree structures in Sections 00–07. This identified problem cases to correct, such as those described in the following section.

### 3.2 Exceptional cases

Firstly, when the lowest spanning node covers the `NML` or `JJP` bracket exactly, no changes need to be made to CCGbank. These cases occur when CCGbank already received the correct structure during the original conversion process. For example, brackets separating a possessive from its possessor were detected automatically.

A more complex case is conjunctions, which do not follow the simple head/adjunct method of assigning supertags. Instead, conjuncts are identified during the head-finding stage, and then assigned the supertag dominating the entire coordination. Intervening non-conjunct nodes are given the same category with the *conj* feature, resulting in a derivation that can be parsed with the standard CCGbank binary coordination rules:

$$conj \;\; X \;\; \Rightarrow \;\; X[conj] \quad (8)$$

$$X \;\; X[conj] \;\; \Rightarrow \;\; X \quad (9)$$

The derivation in Figure 1(b) is produced by these corrections to coordination derivations. As a result, applications of the non-CCG rule shown in (3) have been reduced from 1378 to 145 cases.

Some POS tags require special behaviour. Determiners and possessive pronouns are both usually given the supertag $NP[nb]/N$, and this should not be changed by the conversion process. Accordingly, we do not alter tokens with POS tags of `DT` and `PRP$`. Instead, their sibling node is given the category $N$ and their parent node is made the head. The parent's sibling is then assigned the appropriate adjunct category (usually $NP\backslash NP$). Tokens with punctuation POS tags[1] do not have their supertag changed either.

Finally, there are cases where the lowest spanning node covers a constituent that should not be changed. For example, in the following NP:

```
(NP
  (NML (NN lower) (NN court) )
  (JJ final) (NN ruling) )
```

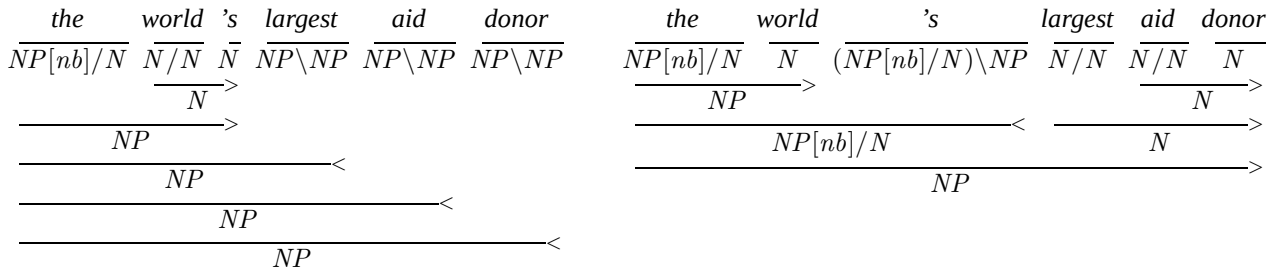
with the original CCGbank lowest spanning node:

```
(N (N/N lower)
  (N (N/N court)
    (N (N/N final) (N ruling) ) ) )
```

the `final ruling` node should not be altered.

It may seem trivial to process in this case, but consider a similarly structured NP: `lower court ruling that the U.S. can bar the use of...` Our minimalist approach avoids reanalysing the many linguistic constructions that can be dominated by NPs, as this would reinvent the creation of CCGbank. As a result, we only flatten those constituents that partially overlap the `NML` or `JJP` bracket. The existing structure and dependencies of other constituents are retained. Note that we are still converting every `NML` and `JJP` bracket, as even in the subordinate clause example, only the structure around `lower court` needs to be altered.

[1] period, comma, colon, and left and right bracket.




(a) (b)

Figure 3: CCGbank derivations for possessives

| | # | % |
|---|---|---|
| Possessive | 224 | 43.75 |
| Left child contains DT/PRP$ | 87 | 16.99 |
| Couldn't assign to non-leaf | 66 | 12.89 |
| Conjunction | 35 | 6.84 |
| Automatic conversion was correct | 26 | 5.08 |
| Entity with internal brackets | 23 | 4.49 |
| DT | 22 | 4.30 |
| NML/JJP bracket is an error | 12 | 2.34 |
| Other | 17 | 3.32 |
| Total | 512 | 100.00 |

Table 1: Manual analysis

### 3.3 Manual annotation

A handful of problems that occurred during the conversion process were corrected manually. The first indicator of a problem was the presence of a possessive. This is unexpected, because possessives were already bracketed properly when CCGbank was originally created (Hockenmaier, 2003, §3.6.4). Secondly, a non-flattened node should not be assigned a supertag that it did not already have. This is because, as described previously, a non-leaf node could dominate any kind of structure. Finally, we expect the lowest spanning node to cover only the NML or JJP bracket and one more constituent to the right. If it doesn't, because of unusual punctuation or an incorrect bracket, then it may be an error. In all these cases, which occur throughout the corpus, we manually analysed the derivation and fixed any errors that were observed.

512 cases were flagged by this approach, or 1.90% of the 26,993 brackets converted to CCG. Table 1 shows the causes of these problems. The most common cause of errors was possessives, as the conversion process highlighted a number of instances where the original CCGbank analysis was incorrect. An example of this error can be seen in Figure 3(a), where the possessive doesn't take any arguments. Instead, *largest aid donor* incorrectly modifies the NP one word at a time. The correct derivation after manual analysis is in (b).

The second-most common cause occurs when there is apposition inside the NP. This can be seen in Figure 4. As there is no punctuation on which to coordinate (which is how CCGbank treats most appositions) the best derivation we can obtain is to have *Victor Borge* modify the preceding NP.

The final step in the conversion process was to validate the corpus against the CCG grammar, first by those productions used in the existing CCGbank, and then against those actually licensed by CCG (with pre-existing ungrammaticalities removed). Sixteen errors were identified by this process and subsequently corrected by manual analysis.

In total, we have altered 12,475 CCGbank sentences (25.5%) and 20,409 dependencies (1.95%).

## 4 NER features

Named entity recognition (NER) provides information that is particularly relevant for NP parsing, simply because entities are nouns. For example, knowing that Air Force is an entity tells us that Air Force contract is a left-branching NP.

Vadas and Curran (2007a) describe using NE tags during the annotation process, suggesting that NER-based features will be helpful in a statistical model. There has also been recent work combining NER and parsing in the biomedical field. Lewin (2007) experiments with detecting base-NPs using NER information, while Buyko et al. (2007) use a CRF to identify

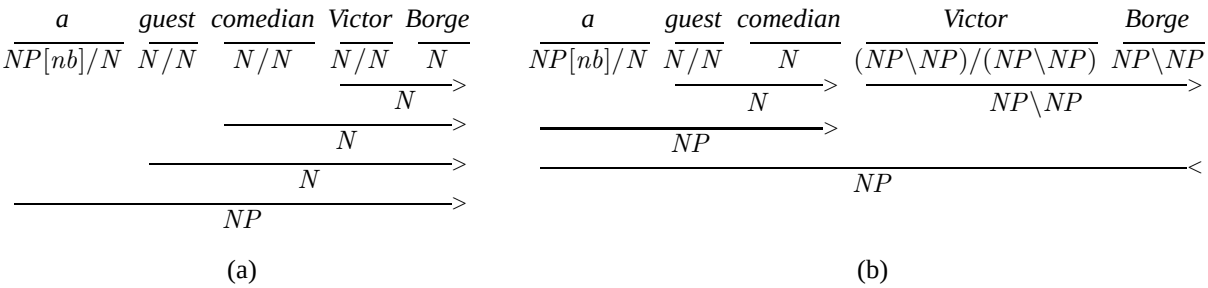



Figure 4: CCGbank derivations for apposition with DT

coordinate structure in biological named entities.

We draw NE tags from the BBN Entity Type Corpus (Weischedel and Brunstein, 2005), which describes 28 different entity types. These include the standard person, location and organization classes, as well person descriptions (generally occupations), NORP (National, Other, Religious or Political groups), and works of art. Some classes also have finer-grained subtypes, although we use only the coarse tags in our experiments.

Clark and Curran (2007b) has a full description of the C&C parser's pre-existing features, to which we have added a number of novel NER-based features. Many of these features generalise the head words and/or POS tags that are already part of the feature set. The results of applying these features are described in Sections 5.3 and 6.

The first feature is a simple lexical feature, describing the NE tag of each token in the sentence. This feature, and all others that we describe here, are not active when the NE tag(s) are **O**, as there is no NER information from tokens that are not entities.

The next group of features is based on the local tree (a parent and two child nodes) formed by every grammar rule application. We add a feature where the rule being applied is combined with the parent's NE tag. For example, when joining two constituents[2]: $\langle$`five`, `CD`, **`CARD`**, $N/N\rangle$ and $\langle$`Europeans`, `NNPS`, **`NORP`**, $N\rangle$, the feature is:

$$N \rightarrow N/N\ N + \textbf{NORP}$$

as the head of the constituent is `Europeans`.

In the same way, we implement features that combine the grammar rule with the child nodes. There are already features in the model describing each combination of the children's head words and POS tags, which we extend to include combinations with the NE tags. Using the same example as above, one of the new features would be:

$$N \rightarrow N/N\ N + \textbf{CARD} + \textbf{NORP}$$

The last group of features is based on the NE category spanned by each constituent. We identify constituents that dominate tokens that all have the same NE tag, as these nodes will not cause a "crossing bracket" with the named entity. For example, the constituent `Force contract`, in the NP `Air Force contract`, spans two different NE tags, and should be penalised by the model. `Air Force`, on the other hand, only spans **`ORG`** tags, and should be preferred accordingly.

We also take into account whether the constituent spans the *entire* named entity. Combining these nodes with others of different NE tags should *not* be penalised by the model, as the NE must combine with the rest of the sentence at some point.

These NE spanning features are implemented as the grammar rule in combination with the parent node or the child nodes. For the former, one feature is active when the node spans the entire entity, and another is active in other cases. Similarly, there are four features for the child nodes, depending on whether neither, the left, the right or both nodes span the entire NE. As an example, if the `Air Force` constituent were being joined with `contract`, then the child feature would be:

$$N \rightarrow N/N\ N + \text{LEFT} + \textbf{ORG} + \textbf{O}$$

assuming that there are more **O** tags to the right.

## 5 Experiments

Our experiments are run with the C&C CCG parser (Clark and Curran, 2007b), and will evaluate the changes made to CCGbank, as well as the effectiveness of the NER features. We train on Sections 02-21, and test on Section 00.

[2]These 4-tuples are the node's head, POS, NE, and supertag.

| | PREC | RECALL | F-SCORE |
|---|---|---|---|
| Original | 91.85 | 92.67 | 92.26 |
| NP corrected | 91.22 | 92.08 | 91.65 |

Table 2: Supertagging results

| | PREC | RECALL | F-SCORE |
|---|---|---|---|
| Original | 85.34 | 84.55 | 84.94 |
| NP corrected | 85.08 | 84.17 | 84.63 |

Table 3: Parsing results with gold-standard POS tags

| | PREC | RECALL | F-SCORE |
|---|---|---|---|
| Original | 83.65 | 82.81 | 83.23 |
| NP corrected | 83.31 | 82.33 | 82.82 |

Table 4: Parsing results with automatic POS tags

| | PREC | RECALL | F-SCORE |
|---|---|---|---|
| Original | 86.00 | 85.15 | 85.58 |
| NP corrected | 85.71 | 84.83 | 85.27 |

Table 5: Parsing results with NER features

### 5.1 Supertagging

Before we begin full parsing experiments, we evaluate on the supertagger alone. The supertagger is an important stage of the CCG parsing process, its results will affect performance in later experiments.

Table 2 shows that F-score has dropped by 0.61%. This is not surprising, as the conversion process has increased the ambiguity of supertags in NPs. Previously, a bare NP could only have a sequence of $N/N$ tags followed by a final $N$. There are now more complex possibilities, equal to the Catalan number of the length of the NP.

### 5.2 Initial parsing results

We now compare parser performance on our NP corrected version of the corpus to that on original CCGbank. We are using the normal-form parser model and report labelled precision, recall and F-score for all dependencies. The results are shown in Table 3.

The F-score drops by 0.31% in our new version of the corpus. However, this comparison is not entirely fair, as the original CCGbank test data does not include the NP structure that the NP corrected model is being evaluated on. Vadas and Curran (2007a) experienced a similar drop in performance on Penn Treebank data, and noted that the F-score for `NML` and `JJP` brackets was about 20% lower than the overall figure. We suspect that a similar effect is causing the drop in performance here.

Unfortunately, there are no explicit `NML` and `JJP` brackets to evaluate on in the CCG corpus, and so an NP structure only figure is difficult to compute. Recall can be calculated by marking those dependencies altered in the conversion process, and evaluating only on them. Precision cannot be measured in this way, as NP dependencies remain undifferentiated in parser output. The result is a recall of 77.03%, which is noticeably lower than the overall figure.

We have also experimented with using automatically assigned POS tags. These tags are accurate with an F-score of 96.34%, with precision 96.20% and recall 96.49%. Table 4 shows that, unsurprisingly, performance is lower without the gold-standard data. The NP corrected model drops an additional 0.1% F-score over the original model, suggesting that POS tags are particularly important for recovering internal NP structure. Evaluating NP dependencies only, in the same manner as before, results in a recall figure of 75.21%.

### 5.3 NER features results

Table 5 shows the results of adding the NER features we described in Section 4. Performance has increased by 0.64% on both versions of the corpora. It is surprising that the NP corrected increase is not larger, as we would expect the features to be less effective on the original CCGbank. This is because incorrect right-branching NPs such as *Air Force contract* would introduce noise to the NER features.

Table 6 presents the results of using automatically assigned POS and NE tags, i.e. parsing raw text. The NER tagger achieves 84.45% F-score on all non-**O** classes, with precision being 78.35% and recall 91.57%. We can see that parsing F-score has dropped by about 2% compared to using gold-standard POS and NER data, however, the NER features still improve performance by about 0.3%.

| | PREC | RECALL | F-SCORE |
|---|---|---|---|
| Original | 83.92 | 83.06 | 83.49 |
| NP corrected | 83.62 | 82.65 | 83.14 |

Table 6: Parsing results with automatic POS and NE tags

| | PREC | RECALL | F-SCORE |
|---|---|---|---|
| Original | 86.86 | 81.61 | 84.15 |
| NP corrected | 87.97 | 82.54 | 85.17 |

Table 7: DepBank gold-standard evaluation

| | PREC | RECALL | F-SCORE |
|---|---|---|---|
| Original | 82.57 | 81.29 | 81.92 |
| NP corrected | 83.53 | 82.15 | 82.84 |
| Original, NER | 82.87 | 81.49 | 82.17 |
| NP corrected, NER | 84.12 | 82.75 | 83.43 |

Table 8: DepBank evaluation results

## 6 DepBank evaluation

One problem with the evaluation in the previous section, is that the original CCGbank is not expected to recover internal NP structure, making its task easier and inflating its performance. To remove this variable, we carry out a second evaluation against the Briscoe and Carroll (2006) reannotation of DepBank (King et al., 2003), as described in Clark and Curran (2007a). Parser output is made similar to the grammatical relations (GRs) of the Briscoe and Carroll (2006) data, however, the conversion remains complex. Clark and Curran (2007a) report an upper bound on performance, using gold-standard CCGbank dependencies, of 84.76% F-score.

This evaluation is particularly relevant for NPs, as the Briscoe and Carroll (2006) corpus *has* been annotated for internal NP structure. With our new version of CCGbank, the parser will be able to recover these GRs correctly, where before this was unlikely.

Firstly, we show the figures achieved using gold-standard CCGbank derivations in Table 7. In the NP corrected version of the corpus, performance has increased by 1.02% F-score. This is a reversal of the results in Section 5, and demonstrates that correct NP structure improves parsing performance, rather than reduces it. Because of this increase to the upper bound of performance, we are now even closer to a true formalism-independent evaluation.

We now move to evaluating the C&C parser itself and the improvement gained by the NER features. Table 8 show our results, with the NP corrected version outperforming original CCGbank by 0.92%. Using the NER features has also caused an increase in F-score, giving a total improvement of 1.51%. These results demonstrate how successful the correcting of NPs in CCGbank has been.

Furthermore, the performance increase of 0.59% on the NP corrected corpus is more than the 0.25% increase on the original. This demonstrates that NER features are particularly helpful for NP structure.

## 7 Conclusion

The first contribution of this paper is the application of the Vadas and Curran (2007a) data to Combinatory Categorial Grammar. Our experimental results have shown that this more accurate representation of CCGbank's NP structure increases parser performance. Our second major contribution is the introduction of novel NER features, a source of semantic information previously unused in parsing.

As a result of this work, internal NP structure is now recoverable by the C&C parser, a result demonstrated by our total performance increase of 1.51% F-score. Even when parsing raw text, without gold standard POS and NER tags, our approach has resulted in performance gains.

In addition, we have made possible further increases to NP structure accuracy. New features can now be implemented and evaluated in a CCG parsing context. For example, bigram counts from a very large corpus have already been used in NP bracketing, and could easily be applied to parsing. Similarly, additional supertagging features can now be created to deal with the increased ambiguity in NPs.

Downstream NLP components can now exploit the crucial information in NP structure.

## Acknowledgements

We would like to thank Mark Steedman and Matthew Honnibal for help with converting the NP data to CCG; and the anonymous reviewers for their helpful feedback. This work has been supported by the Australian Research Council under Discovery Project DP0665973.

## References


Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Ted Briscoe 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, Australia.

Ekaterina Buyko, Katrin Tomanek, and Udo Hahn. 2007. Resolution of coordination ellipses in biological named entities with conditional random fields. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING-2007)*, pages 163–171. Melbourne, Australia.

Stephen Clark and James R. Curran. 2007a. 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, Czech Republic.

Stephen Clark and James R. Curran. 2007b. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

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

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

Julia Hockenmaier and Mark Steedman. 2005. CCGbank manual. Technical Report MS-CIS-05-09, Department of Computer and Information Science, University of Pennsylvania.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Matthew Honnibal and James R. Curran. 2007. Improving the complement/adjunct distinction in CCGbank. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING-07)*, pages 210–217. Melbourne, Australia.

Tracy Holloway King, 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)*. Budapest, Hungary.

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*, pages 47–54. Cambridge, MA.

Ian Lewin. 2007. BaseNPs that contain gene names: domain specificity and genericity. In *Biological, translational, and clinical language processing workshop*, pages 163–170. Prague, Czech Republic.

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. 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.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

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

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 of Computational Linguistics (ACL-07)*, pages 240–247. Prague, Czech Republic.

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)*, pages 104–112. Melbourne, Australia.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical report.