Statistical Parsing for CCG with Simple Generative Models

Julia Hockenmaier
Division of Informatics
University of Edinburgh
julia@cogsci.ed.ac.uk

Abstract

This paper presents a statistical parser for a wide-coverage Combinatory Categorial Grammar (CCG) derived from the Penn Treebank. The Treebank is translated to a corpus of canonical CCG derivations. We define a generative statistical model over CCG derivations and train it on the transformed Treebank. This model is evaluated using Parseval measures and the accuracy of recovery of word-word dependencies. The impact of lexical coverage on parsing accuracy is also investigated.

1 Introduction

Combinatory Categorial Grammar (CCG, Steedman (2000)) is a mildly context-sensitive, lexicalized grammar formalism that offers transparent semantics, as well as a simple, monotonic account of the long-range dependencies and fragmentary constituents that arise in coordinate constructions, extraction phenomena, and prosodic phrasing. It therefore has potential advantages for wide-coverage parsing of spoken and written text, in which some of these constructions are very common. Despite some preliminary work (Doran and Srinivas, 1994; Srinivas, 1997), these advantages have not yet been fully realized.

As the currently most successful probabilistic wide-coverage parsers (eg. Collins (1998), Charniak (1999)) are all based on models acquired from the Penn Treebank (Marcus et al., 1993), their underlying grammars lack an adequate treatment of coordination and extraction, and do not yield interpretable structures.

The present paper seeks to bring these two lines of inquiry together by integrating probabilistic wide-coverage parsing with a linguistically well-informed treatment of coordination and long-range dependencies for which semantic representations can be built during parsing.

While this paper is only concerned with syntax, Bierner (2001) shows how simple predicate-argument structure semantics can be acquired for categorial lexica, an approach which can easily be applied for our purpose.

As a first step towards statistical parsing with CCG, we have translated the Penn Treebank to a corpus of canonical CCG derivation trees, which serves as our training and test data. Section 3 describes the translation procedure.

In section 4, we present a very simple generative model over CCG derivation trees. This model uses much less explicit lexical information than current state-of-the-art Treebank parsers such as Collins (1998) and Charniak (1999). However, as this is the first attempt at statistical parsing with CCG, our present aim is to establish a baseline performance, comparable to simple PCFGs.

The performance of the model is described and evaluated in section 6. Standard Parseval measures on trees are not the most appropriate evaluation metric for categorial derivation trees. In contrast to context-free grammars extracted from the Penn Treebank, there can be many different, but equivalent, binary categorial trees. Hence, the rate of recovery of tree nodes is an unduly harsh measure of performance. Collins (1998) evaluates the attachment accuracy of his parser by reporting its accuracy in recovering word-word dependencies. Although his definition of lexical dependencies does not yield all "true" semantic dependencies, it gives us an evaluation metric which is neutral which respect to equivalent branchings of binary trees. This problem of residual "spurious" ambiguities will be discussed in section 6.3.

The lexicon plays a very important role in CCG, and we show that the coverage of the CCG lexicon acquired from the Treebank has a big impact on the parsing accuracy of the model. We conclude this paper with a discussion of our results, and an outlook on how they could be improved upon by more sophisticated models.

2 Combinatory Categorial Grammar

We only give a brief introduction here, and refer the reader to Steedman (2000) for a detailed introduction to CCG. In categorial grammar, information about word order and valence is encoded directly in syntactic categories which are assigned to words. These syntactic categories specify the number of arguments a word can take, as well as the relative position of arguments with respect to the head. For instance, the category of the transitive verb **loves** is as follows

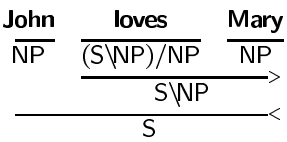
(1) **loves** $\vdash$ (S\NP)/NP

A syntactic category is also paired with a semantic interpretation, such as $\lambda x.\lambda y.loves(y,x)$, but in this paper we are only concerned with syntactic derivations.

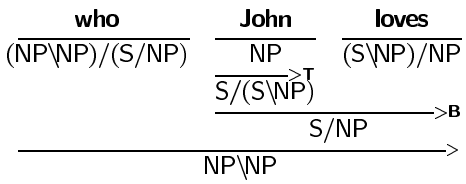
Constituents can combine via a set of combinatory rules, which are stated as schemata over categories (forward application, composition and type-raising in the following example):

(2)
X/Y	Y	$\Rightarrow$	X	(>)
X/Y	Y/Z	$\Rightarrow$	X/Z	(>**B**)
X		$\Rightarrow$	T/(T\X)	(>**T**)

The normal-form derivation of ordinary sentences such as *John loves Mary* only requires function application:

(3)


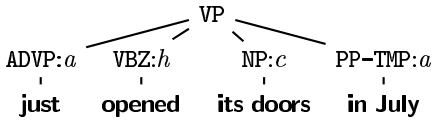
Composition and Type-raising are syntactically necessary to capture long-distance dependencies and "non-constituent" coordination:

(4)


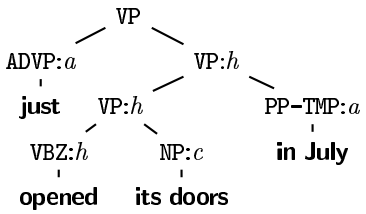
3 Translating the Treebank

The Wall Street Journal sub-corpus of the Penn Treebank contains 1 million words of parsed and tagged Wall Street Journal text collected in 1989. The markup is designed to allow a distinction between complements and adjuncts. In Hockenmaier et al. (2000), we presented a procedure for extracting a categorial lexicon from the Treebank. However, in order to obtain a corpus of categorial derivation trees, the flat Treebank trees have also to be transformed to binary trees.

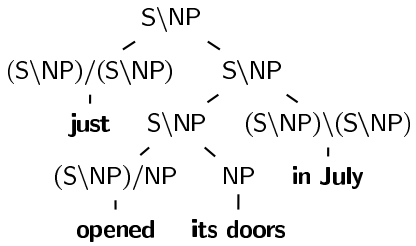
The translation process works in the following manner: first, the constituent type of each node (head (h), complement (c), or adjunct (a)) is identified, using a method similar to Collins (1998):

(5)


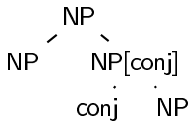
Then the flat trees are transformed to binary trees:

(6)


Categories are assigned to the nodes in a binary tree in the following recursive, top-down manner: The category of the root is determined by its Treebank label (eg.$\{\texttt{S},\texttt{SINV},\texttt{SQ}\} \rightarrow$ S). The category of a complement child is defined by a similar mapping from Treebank labels to the atomic categories. Given a parent category X, the category of an adjunct child is a unary functor X/X if the adjunct child is the left daughter, or X\X if it is the right daughter. If the non-head daughter is an adjunct, the category of the head daughter is identical to the category of the parent. If the non-head daughter is a complement with category Y, the head daughter is either X/Y or X\Y, depending on its position.

(7)


The treatment of conjunction When transforming trees with coordinate constructions (or lists) to binary trees, the dummy node which is inserted receives the same category as both conjuncts, but additionally carries a feature [conj]. A node with category X[conj] always has its head daughter (with category X) on the right, and the left daughter is either a conjunction (conj), or a punctuation mark, such as a comma or semicolon:

(8)


Alterations to Treebank analyses In order to obtain the desired categorial analyses, a few alterations to the original Treebank analyses are made. For instance, a noun level is inserted in the otherwise flat NP structure in the Treebank.

The treatment of null elements In order to obtain the correct analysis of long-distance dependencies (see example 4), the presence of `*T*` traces in the Treebank is exploited: during category assignment, `*T*` traces are treated as ordinary complements, and a mechanism similar to slash-passing is used to guarantee that the category of a sentence out of which an element has been extracted is correct. See Hockenmaier et al. (2000) for details on the treatment of this and other null elements by the translation algorithm.

4 A Simple Generative Model

In this section, we describe a basic generative model over CCG derivation trees. This model does not incorporate the notion of combinatory rules. Instead, it is a general model over binary and unary trees, and it is only by virtue of the fact that it is trained on CCG derivation trees that it becomes a model of CCG.

Probabilistic generative models are built upon the assumption that the data is generated by a stochastic process. When defining generative models for language, it is customary to tie the process to the grammar rules, as this guarantees that only strings generated by the grammar receive a non-zero probability. Instead of viewing a CCG derivation as a bottom-up process, we assume that the underlying process is a top-down tree generator, starting with a root node of category S. The model is defined in terms of probability distributions over local trees (trees of depth 1). A node N with category X can expand in different ways: either it is a leaf node (in which case the expansion stops), or it has one or two daughters. This is expressed by the **expansion probability** $P(exp \mid \mathsf{X})$. If N is a leaf node, a word **w** is generated (with the **lexical probability** $P(\mathbf{w} \mid \mathsf{X}, exp = leaf)$). If N is not a leaf node, we have to generate its daughters. Previous work on statistical parsing (eg. Collins (1998)) has shown that lexical dependency information is important for parse disambiguation. Although the basic model does not yet capture lexical dependencies, we wish to be able to extend it easily to do so. Therefore, the model distinguishes head daughters from non-head daughters (such as complements or adjuncts, although this information does not need to be made explicit in the model, as it is captured by the categories themselves). Also, *exp* distinguishes between binary trees where the head daughter is right and left. Thus, we first generate the category of the head daughter H with the **head probability** $P(\mathsf{H} \mid \mathsf{X}, exp)$. If N has two daughters (*exp* is `left` or `right`), we also generate the category of the non-head daughter D (with the **non-head probability** $P(\mathsf{D} \mid \mathsf{X}, exp, \mathsf{H})$). Then H and D are expanded themselves. To summarize, the model contains the following families of probability distributions:

Expansion probability $P(exp \mid \mathsf{X})$**:** Possible values of *exp*: {`leaf`, `unary`, `left`, `right`}.

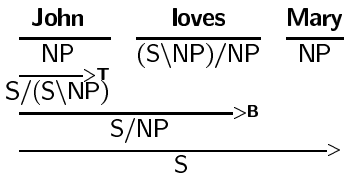
Head probability $P(\mathsf{H} \mid \mathsf{X}, exp)$**:** The category H of the head daughter. Only for $exp \neq$ `leaf`

Non-head probability $P(\mathsf{D} \mid \mathsf{X}, exp, \mathsf{H})$**:** The category D of the sister daughter. Only for $exp =$ `left` and $exp =$ `right`

Lexical probability $P(\mathbf{w} \mid \mathsf{X}, exp =$ `leaf`$)$**:** The word **w**. Only for $exp =$ `leaf`

The probability of a local tree is the product of the relevant probabilities, and the probability of a tree is the product of the probabilities of each local tree within this tree.

The importance of "canonical" CCG derivations Because of CCG's treatment of extraction, even simple sentences, such as *John loves Mary* can have multiple derivations; in addition to (3), the following is also allowed:

(9)


As the translation procedure described in section 3 maintains the original Treebank bracketing wherever possible, the obtained CCG derivations correspond to normal form analyses which use function application wherever possible. Type-raising and composition, which are required for these "non-standard" analyses, are only used when required by syntax, ie. in the presence of extraction or so-called "non-constituent" coordination. This is very important for the model described in this section, as otherwise the probability mass assigned to a simple sentence such as *John loves Mary* would be split between two derivations. *John* and *loves* can still combine to a constituent S/NP. However, in terms of the tree-generating process, the constituent S/NP is only generated in the presence of extraction (cf. (4)), and therefore receives zero probability otherwise.

	LexCat	LP	LR	BP	BR	CM	CB	0 CB	≤2 CB
Total	86.51%	67.78%	67.57%	73.61%	73.38%	6.05%	5.30	18.48%	37.90%
≤40 words	86.77%	68.74%	68.56%	74.58%	74.37%	6.50%	4.66	19.85%	40.44%
Missing entries	79.37%	57.90%	57.95%	67.02%	67.15%	0.15%	8.22	5.59%	18.24%
No missing	90.21%	72.95%	72.57%	77.00%	77.41%	8.45%	4.11	23.74%	45.92%

Figure 1: Accuracy of lexical categories and Parseval measures (see sections 6.1 and 6.2)

5 The Experiment

We trained a model on sections 02-21 of the translated WSJ Treebank, and tested it on section 23. The test corpus has 2373 sentences, out of which 2360 sentences could be parsed. The grammar acquired from the training corpus has 1251 lexical categories and 3493 grammar rules, out of which 1320 appear only once. On average, a word has 1.77 lexical categories. We use a simple bottom-up chart parser, and compute inside probabilities during the parse. A beam search similar to Collins (1998) is used for efficiency reasons.

5.1 Treatment of unknown words

An important factor in the performance of a statistical parser is the treatment of unknown words. Collins (1998) replaces all words which occur less than 5 times in the training data by the token `UNKNOWN`, which is then treated like an ordinary word. Any unknown word in the test data is also replaced by this token, and the tag given by a POS-tagger for this word is used as the only possible tag for that word. We also estimate the probabilities for unknown words by estimating over rare words in the training data. But as we do not have a CCG tagger, replacing all rare words by the same token would mean that we would have to assign all lexical categories seen with rare words in the training data to unknown words in the test data. Given that the number of lexical categories is so high, this is not feasible. In order to get a less crude estimate, we replace rare words in the training corpus (words with a frequency ≤ 3 in WSJ 02-21) by their part-of-speech tags, and estimate the lexical probabilities for unknown words separately for each part of speech. When parsing, we assume (like Collins) that the input is tagged, so that we can use the POS tag of an unknown word to obtain its lexical probabilities.

6 Results

6.1 Parseval measures

Figure 1 gives the accuracy of lexical categories as well as the standard Parseval measures labelled precision and recall, bracketed precision and recall, the percentage of sentences with complete match, the average number of crossing brackets, the percentage of sentences with no crossing brackets, and the percentage of sentences with two or fewer crossing brackets. The first two rows give overall performance, and performance on sentences of 40 words or less.

6.2 Coverage of the acquired lexicon

As CCG is a lexicalized formalism, in which much of the syntactic information is given by the lexicon, coverage of the lexicon largely determines coverage of the grammar. For around 1.8% of the tokens in section 23 the correct lexical entry is not in the categorial lexicon acquired from sections 02-21, although the word has been seen in the training corpus. When parsing with this lexicon, this means that the sentences in which such words occur will either get an incorrect or no parse. This also accounts for the 13 test sentences which could not been parsed. Out of the 2360 test sentences which could be parsed, 692 require categories missing in the lexicon. Therefore, Figure 1 gives separate figures for the sentences in which at least one lexical entry is missing, and for the sentences in which all correct word-category pairs are in the lexicon (last row). Performance for sentences with missing entries is much lower.

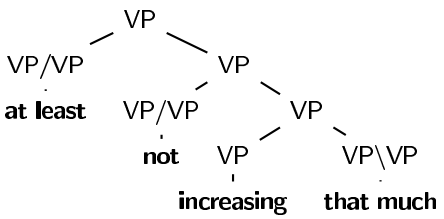
6.3 Residual spurious ambiguities

Standard Parseval is unduly harsh on binary categorial trees for two reasons: firstly, measures such as crossing brackets penalize binary trees, which have many more bracketings than flat Treebank trees. Therefore, our results are not directly comparable to parsers which aim to reconstruct the original Treebank structures. Perhaps more importantly, despite the use of a normal-form grammar, there are situations where there are different equivalent binary bracketings, and it is only by chance that the model picks the one which occurs in the reference. Consider the following example:

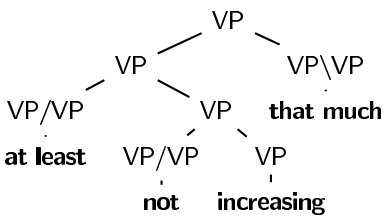
(10)
```
(VP (ADVP (IN at) (JJS least))
    (RB not)
    (VBG increasing)
    (ADVP (RB that) (RB much)))
```

There are three different, equivalent categorial derivations for the same string [1]:

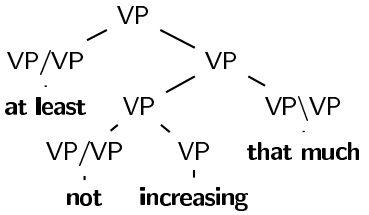
(11) a.



b.



c.



Only the first of these appears in the reference corpus, and the other two trees would obtain precision and recall scores of only 33.3%, although they are assigned the same probability mass according to the present model. There are different ways in which this problem could be dealt with: we could incorporate a distance measure which favours the canonical bracketing (similar to the distance measure used by Collins (1998), which favours right-branching trees, but for binary trees). Another solution might be to produce flat trees (eg. by merging equivalent binary branchings) in which this problem does not arise. Or we can use an evaluation metric which does not penalize these spurious ambiguities. Collins (1998) evaluates the attachment accuracy of his parser by examining the recovery of word-word dependencies as defined by the parse trees. In this approach, a dependency relation *rel* is defined as a triple <P,H,D>, where P, H and D are the labels of the parent node, the head daughter, and the non-head daughter in a local tree. $rel(w_i, w_j)$ holds if w_i is the head word of H and w_j the head word of D[2] A binary tree spanning a string of n words has $n-1$ non-leaf nodes, each of which determines one dependency relation. Although not fully equivalent, we can therefore view this dependency evaluation measure on our binary trees as a metric which is neutral with respect to equivalent binary branchings. There is an additional relation $head(w_i)$ for the sentential head.

[1] We use VP to abbreviate S\NP

[2] Collins also has a directional parameter, *dir*, which we do not need as directionality is implicit in the categories.

6.4 Dependency accuracy

Figure 2 gives the dependency accuracy. The second column gives overall accuracy, including $head(w_i)$. The fourth column gives accuracy of the dependency relation <P,H,D>. The fifth column defines a *rel* as <D> only. In the last column relational labels are disregarded, and accuracy of $rel(w_i, w_j)$ holds if w_j is a modifier of w_i. As can be seen from Figure 2, these figures are indeed much higher than the Parseval scores.

Again, coverage of the lexicon has a big impact on performance: Figure 2 shows that labelled dependency recovery is about 15.6% higher for sentences where all correct entries are in the lexicon.

7 Discussion

This paper presents a first attempt at wide-coverage statistical parsing with CCG, and aims to establish the baseline performance which can be obtained with a very simple model. We conjecture that this baseline performance can be improved upon by including additional parameters such as word-word dependencies or distance measures on the surface string. At a first glance, our performance seems low in comparison to unlexicalized PCFGs estimated from the original Treebank: Charniak (1996) reports 78.8% bracketed precision and 80.4% bracketed recall on sentences of up to 40 words. However, in that experiment all words are replaced with their correct part-of-speech tags. Our low performance seems to be largely due to the coverage of the lexicon. In fact, for the subcorpus for which we have complete coverage, our bracketed precision and recall scores are not much lower with 77.0% and 77.41%, despite the fact that standard Parseval measure are overly strict on binary categorial trees. Also, performance as measured by the recovery of dependencies is higher, and on the subcorpus with complete coverage reaches 86.51%, which is only 4.5 percent lower than Collins (1998), who gives 91.0% recall of unlabelled dependencies. This result is even more striking given how naive our current model is in comparison to Collins' models.

For standard Treebank parsing approaches, a lot of improvement in performance has been obtained through lexicalization of the models, that is, by using lexical items as conditioning variables. It is unclear whether this would yield a similar improvement for a lexicalized *grammar formalism* such as CCG. It is likely that lexicalization of

	#sent	All	$head(w_i)$	rel =<P,H,D>	rel =<D>	Unlabelled Match
All	2360	74.35%	90.04%	73.55%	79.22%	82.83%
Missing entries	692	63.72%	80.20%	62.97%	70.18%	75.88%
No missing entry	1668	79.32%	94.12%	79.16%	84.01%	86.51%

Figure 2: Dependency evaluation (see section 6.4)

the *derivational probabilities* is not necessary, as CCG categories are much more informative than the original Treebank labels. However, it is to be expected that the improvement obtained from incorporating *word-word dependencies* will carry over to a CCG approach, and this is an area where using CCG might give us a distinct advantage over conventional PSG approaches: CCG's succinct analysis of coordination and long-distance dependencies means that more lexical dependencies can be captured than has so far been practicable in PSG approaches, and it is likely that this will further improve performance. Other improvements are possible. For instance, we do not yet exploit information given by punctuation marks, conjunctions, or other features of the surface string, which have all been found to improve performance for other Treebank parsers.

Another open question is the choice of the most appropriate evaluation procedure. The standard Parseval metrics have been criticized for many different reasons, and we believe that the recovery of the "true" semantic dependencies, or grammatical roles, as presented by Carroll et al. (1999) is a much better evaluation metric. We have not used this metric for our work yet, as it requires a mapping from our categorial labels to a set of predefined grammatical relations.

The results presented in this paper illustrate the problem of lexical coverage. Because of CCG's large category set, there is not only the usual problem of unknown words, but we also do not necessarily observe the complete set of lexical categories for each word in the training data. This problem does not arise to the same degree in grammars with small category sets, such as the grammar underlying the Penn Treebank. We are currently investigating how lexical rules can be induced for the acquired lexicon, and how CCG supertagging might help. We believe that this will greatly improve coverage and accuracy of any statistical CCG parser.

8 Acknowledgements

The research was funded in part by the Edinburgh Language Technology Group, an EPSRC studentship and EPSRC grant GR/M96889.

I would like to thank Mark Steedman, Stephen Clark and Miles Osborne for many helpful discussions.

References

Gann Bierner. 2001. *Alternative phrases: theoretical analysis and practical applications.* Ph.D. thesis, Division of Informatics, University of Edinburgh.

John Carroll, G. Minnen, and E. Briscoe. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL-99 Workshop on Linguistically Interpreted Corpora (LINC-99).*

Eugene Charniak. 1996. Tree-bank grammars. Technical Report CS-96-02, Department of Computer Science, Brown University.

Eugene Charniak. 1999. A maximum-entropy-inspired parser. Technical Report CS-99-12, Department of Computer Science, Brown University.

Michael Collins. 1998. *Head-Driven Statistical Models for Natural Language Parsing.* Ph.D. thesis, University of Pennsylvania.

Christine Doran and Bangalore Srinivas. 1994. A wide coverage CCG parser. In Anne Abeillé and Owen Rambow, editors, *Proceedings of the 3rd TAG+ workshop, Jussieu.* CSLI, Stanford CA.

Julia Hockenmaier, Gann Bierner, and Jason Baldridge. 2000. Providing Robustness for a CCG System. In *Proceedings of ESSLLI'2000 Workshop on Linguistic Theory and Grammar Implementation.*

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational linguistics*, 19:313–330.

Bangalore Srinivas. 1997. *Complexity of Lexical Descriptions and Its Relevance to Partial Parsing.* Ph.D. thesis, University of Pennsylvania. IRCS Report 97-10.

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