Acquiring Compact Lexicalized Grammars from a Cleaner Treebank

Julia Hockenmaier and Mark Steedman

Division of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW
United Kingdom
{julia,steedman}@cogsci.ed.ac.uk

Abstract

We present an algorithm which translates the Penn Treebank into a corpus of Combinatory Categorial Grammar (CCG) derivations. To do this we have needed to make several systematic changes to the Treebank which have to effect of cleaning up a number of errors and inconsistencies. This process has yielded a cleaner treebank that can potentially be used in any framework. We also show how unary type-changing rules for certain types of modifiers can be introduced in a CCG grammar to ensure a compact lexicon without augmenting the generative power of the system. We demonstrate how the combination of preprocessing and type-changing rules minimizes the lexical coverage problem.

1. Introduction

Expressive lexicalized grammar formalisms such as HPSG, LFG, TAG and CCG yield richer and semantically more accurate structural representations than plain phrase structure grammars. However, when creating resources for these grammar formalisms from corpora like the Penn Treebank (see related work on the extraction of LTAGs (Xia, 1999; Chen and Vijay-Shanker, 2000) and LFG F-structures (Frank et al., forthcoming) from treebanks), inaccuracies and inconsistencies in the annotation present a more serious problem than they do for approaches which rely directly on the backbone phrase structure grammar. The reason is that noise in the data engenders non-compact grammars and further exacerbates the problem of unseen word-category combinations that arises because of the rich category set these formalisms assume.

We present an algorithm which translates the Penn Treebank into a corpus of Combinatory Categorial Grammar (CCG) derivations. This is an extension of an algorithm to extract a CCG lexicon from the Penn Treebank, presented in Hockenmaier et al. (2002). The basic algorithm is a simple recursive, top-down procedure. However, in order to obtain the desired categorial derivation trees, some systematic changes on the original Treebank trees need to be performed. This preprocessing step also corrects a number of inconsistencies and annotation errors in the data. As a result, we create a cleaner version of the original Penn Treebank which we hope will also be usable for other approaches and grammar formalisms.

Furthermore, we show how unary type-changing rules for certain types of adjuncts can be introduced into the grammar to ensure a compact lexicon without augmenting the generative power of the system. This demonstrates that a wide coverage CCG does not require a prohibitively large lexicon.

2. Combinatory Categorial Grammar

We give only a brief introduction to CCG here, referring the reader to Steedman (2000) for more detail. In categorial grammar, information about word order and valency 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 *bought* is as follows:

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

A syntactic category is also paired with a semantic interpretation, such as $\lambda x.\lambda y.buy(y,x)$, but in this paper we are only concerned with syntactic derivations. In addition to standard function application ((>) below), CCG allows constituent to combine via a set of combinatory rules, which are stated as schemata over categories (forward composition (>**B**) and forward type-raising (>**T**) in the following example):

$$\begin{array}{llcll}
\mathsf{X/Y} & \mathsf{Y} & \Rightarrow & \mathsf{X} & (>) \\
\mathsf{X/Y} & \mathsf{Y/Z} & \Rightarrow & \mathsf{X/Z} & (>\mathbf{B}) \\
\mathsf{X} & & \Rightarrow & \mathsf{T/(T\backslash X)} & (>\mathbf{T})
\end{array}$$

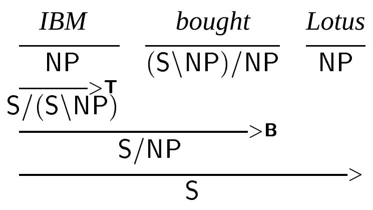
The normal-form derivation of ordinary sentences such as *IBM bought shares* only requires function application:

(2)
$$\begin{array}{ccc}
\textit{IBM} & \textit{bought} & \textit{Lotus} \\
\hline
\mathsf{NP} & \mathsf{(S\backslash NP)/NP} & \mathsf{NP} \\
 & \multicolumn{2}{c}{\overline{\quad\mathsf{S\backslash NP}\quad}^{>}} \\
\multicolumn{3}{c}{\overline{\quad\quad\mathsf{S}\quad\quad}^{<}}
\end{array}$$

Composition and Type-raising are syntactically necessary to capture argument-cluster coordinations (see section 5) and long-distance dependencies:

(3)
$$\begin{array}{ccc}
\textit{what} & \textit{IBM} & \textit{bought} \\
\hline
\mathsf{(NP\backslash NP)/(S/NP)} & \mathsf{NP} & \mathsf{(S\backslash NP)/NP} \\
 & \overline{\mathsf{S/(S\backslash NP)}}^{>\mathbf{T}} & \\
 & \multicolumn{2}{c}{\overline{\quad\mathsf{S/NP}\quad}^{>\mathbf{B}}} \\
\multicolumn{3}{c}{\overline{\quad\quad\mathsf{NP\backslash NP}\quad\quad}^{>}}
\end{array}$$

The construction of non-standard constituents of type S/NP corresponding to the residues of relativization is perfectly general, extending to unbounded or recursive traces, such as $[_{NP}\,\textit{which analysts predict that IBM will buy}]$. However, the presence in the grammar of constituents of this kind allows alternative derivations for sentences like example 2 such as the following:

(4)


Such alternative derivations are grammatically impeccable, since the semantics of combinatory rules guarantees them to deliver the same predicate-argument relations as traditional derivations like example 2. They are sometimes for this reason referred to (misleadingly) as "spurious" ambiguities. For the parser, though, they present a problem, as they engender a potentially exponential growth in the search space.

There are two standard techniques for eliminating this problem while retaining the linguistic advantages of flexible constituency. The first is logical form subsumption-checking table-driven parsing (Karttunen (1989)), in which a chart of canonical forms such as old-style deep structures is maintained, and on deriving a new constituent, before that constituent is added to the chart the parser first checks that such a form is not already there. If it is, the newly derived one can safely be discarded, since all rules of CCG depends solely on categorial types, not on derivation.

The other technique is "normal form" parsing (Hepple and Morrill (1989); König (1994); Eisner (1996)), according to which derivations including type-raising and composition are only used as a last resort, in cases where there is no other way to get an analysis, as in example 3.

In order to build parsers of both kinds, we needed a treebank exhibiting the predicate-argument relations delivered by both. For the normal form parser (Hockenmaier and Steedman (to appear)) it is useful if the trees in the treebank are themselves normal forms. For the other kind of parser (Clark et al. (to appear)) the underlying dependencies can be recovered from them. Such trees can also be readily transduced into other forms suitable for use with other grammatical formalisms. Furthermore, the normal-form derivation arises naturally from the translation procedure described below. We therefore chose to build a normal-form CCG treebank.

3. The Penn Treebank

The Wall Street Journal subcorpus of the Penn Treebank contains around 1 million words of parsed and tagged Wall Street Journal text collected in 1989.

Since the Penn Treebank annotation scheme is designed to encode the underlying predicate-argument structure, it is intended to allow for a clear distinction between arguments and adjuncts. However, this distinction is not always marked explicitly. Therefore we use a heuristic procedure which relies on the label of a node and its parent to determine whether a node is a complement or a modifier. Syntactic heads are also not indicated explicitly, but we adapted the head-finding procedure originally given by Magerman (1994) to our purposes.

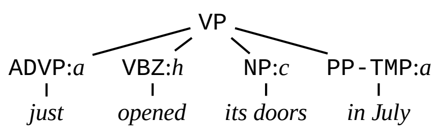
In addition, the Treebank markup contains different types of null elements encoding traces, multiple attachments and attachment ambiguities. The presence of these null elements makes it possible to translate the Treebank trees to the corresponding CCG derivations for relative clauses, wh-questions and coordinate constructions such as right-node raising. Before explaining how this can be done, we present the basic algorithm.

4. Translating the Penn Treebank to CCG

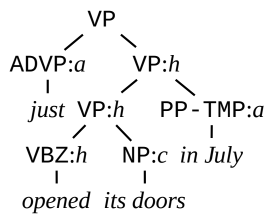
The basic algorithm for translating the Penn Treebank to CCG consists of three steps, each of which is a simple top-down recursive procedure:

```
foreach tree $\tau$:
    determineConstituentType($\tau$);
    makeBinary($\tau$);
    assignCategories($\tau$);
```

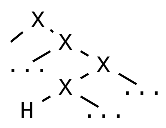
First, the constituent type of each node (head (*h*), complement (*c*), or adjunct (*a*)) is determined, using a method adapted from Magerman (1994) and Collins (1998).



Then the flat trees are transformed to binary trees.



This binarization process inserts dummy nodes into the tree such that all children to the left of the head branch off in a right-branching tree, and then all children to the right of the head branch off in a left-branching tree:



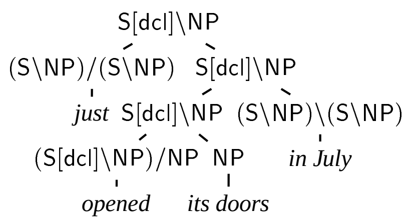
Categories are assigned to the nodes in a binary tree in the following manner (corresponding to a reverse CCG derivation):

- The category of the *root* node is determined by its Treebank label (eg.{S, SINV, SQ} → 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. The fact that English CCG allows the combinatory rules of forward (non-crossing) composition and backward (crossing and non-crossing) composition allows us to strip off complements of X when assigning categories to modifiers. This means for example

that a postverbal modifier within a verb phrase will be (S\NP)\(S\NP), even if its parent does not have the category S\NP, but (S\NP)/NP or similar.

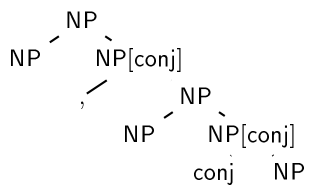
- The category of a *punctuation mark* is the punctuation mark itself.
- The *head* child of a parent with category X has category X if the non-head child is an adjunct or a punctuation mark. If the non-head child is a complement with category Y, the category of the head child is X/Y if the head child is left, and X\Y if the head child is right.

Here is the previous tree annotated with CCG categories:



We use atomic feature values to distinguish different types of sentential and verbal categories (eg. (S[dcl]\NP)/NP): [dcl] for declarative verb forms, [b] for bare infinitives (also for subjunctives and imperatives), [ng] for present participles, [pt] for past participles used in active mode, [pss] for past participles used in passive mode, [em] for embedded declarative sentences, [qem] for embedded questions, [wh] for wh-questions, [q] for yes-no questions, [tpc] for sentences with topicalized noun or verb phrases and [frg] for sentence fragments. The [dcl], [b], [ng], [pt] and [pss] features are determined from the POS-tags of the head verbs and the presence of passive traces. [em], [qem], [q], [frg] and [tpc are determined from the nonterminal labels of the trees. Predicative adjectives are S[adj]\NP. Adjunct categories other than modifiers of adjectives do not carry features.

The treatment of coordination Coordinate constructions (or lists) are transformed into strictly right-branching binary trees. The inserted dummy nodes receive the same category as both conjuncts, but additionally carry 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:

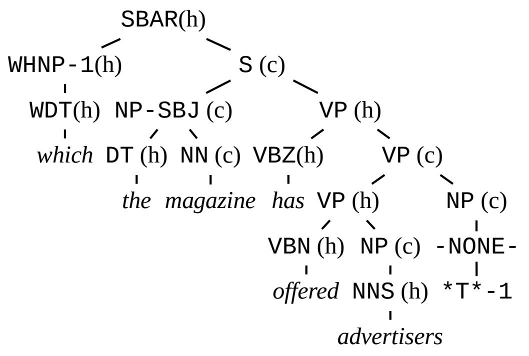


The treatment of null elements In order to obtain the correct analysis of long-distance dependencies, the presence of *T* traces in the Treebank is exploited: during category assignment, *T* traces which stand for extracted arguments are treated as ordinary complements, and a mechanism similar to GPSG's slash-passing (Gazdar et al., 1985) is used to guarantee that the category of a sentence from which an argument has been extracted is correct.

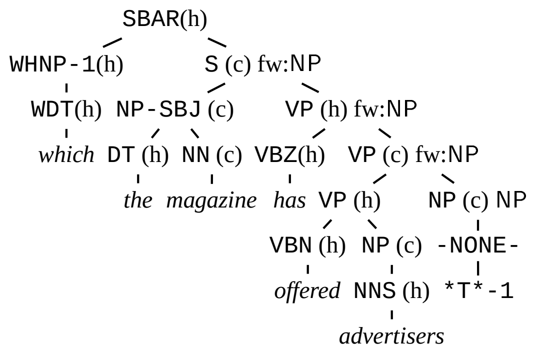
Here is the Treebank analysis of a relative clause:

```
(SBAR (WHNP-1 (WDT which))
  (S (NP-SBJ (DT the) (NN magazine))
     (VP (VBZ has)
         (VP (VBN offered)
            (NP (NNS advertisers))
            (NP (-NONE- *T*-1)))))))
```

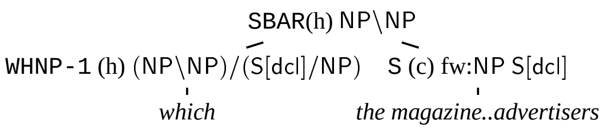
This is the same tree binarized and marked up with the constituent type:



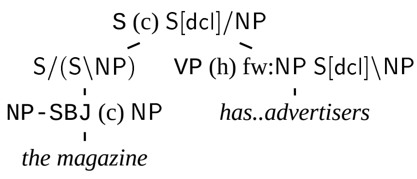
The NP-node with a *T* null element underneath it is a complement trace. The category of a complement trace (here NP) is determined (from its label) before the recursive category assignment described above. This category is then percolated up the head path of the parent of the trace to the next clausal projection. Depending on the position of the trace node within its local tree, this category is marked as a forward (fw) or backward (bw) argument:



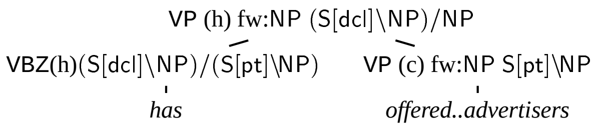
The S node is a complement, and receives category S[dcl]. However, since the S node also has a (forward) trace NP which is not percolated further to its parent, the head daughter (WHNP) subcategorizes for the unsaturated S[dcl]/NP. Assuming that the category of the SBAR parent is NP\NP, the WHNP has category (NP\NP)/(S[dcl]/NP):



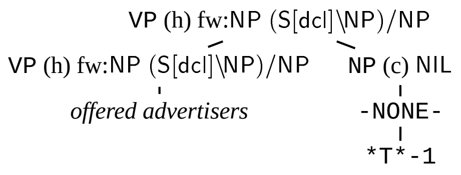
In the next step, the children of the S-node are assigned categories. The NP-SBJ is a backward complement with category NP, but since the S-node has a forward trace, it is type-raised to S/(S\NP). (Like adjunct categories, type-raised categories do not carry features). The forward trace is also appended to the category of the parent S node to yield S[dcl]/NP:



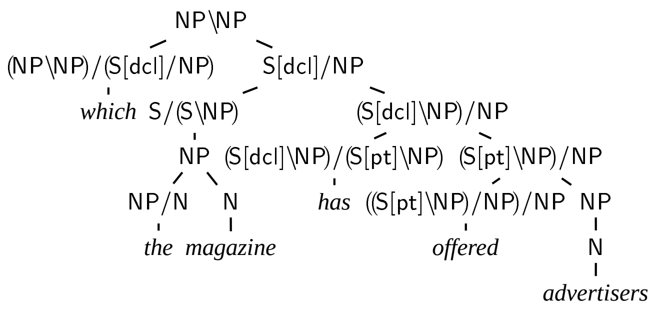
Then, categories are assigned to the children of the VP-node. The VP daughter is a complement with category S[pt]\NP. Since the forward trace is carried up to the parent VP, the head daughter VBZ subcategorizes for S[pt]\NP and hence receives category (S[dcl]\NP)/(S[pt]\NP). The forward trace is also appended to the category of the parent VP:



Going down to the children of the VP daughter, the NP trace is a forward argument. This yields the transitive verb category (S[pt]\NP)/NP for the VBN node. However, since the NP is a trace, this node will be cut out of the tree in a postprocessing stage. The category of the VP parent node is now also changed to take its forward trace into account.



After cutting out the trace and all unary projections X ⇒ X, the subtree of the relative clause is as follows:

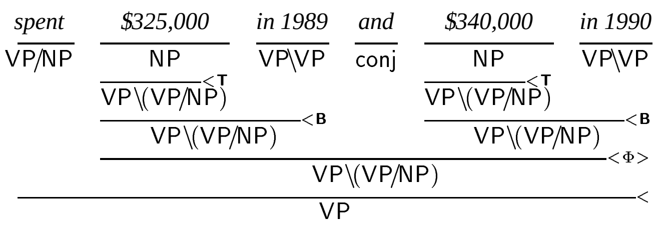


A similar mechanism is used for right-node-raising null elements (*RNR*). See Hockenmaier et al. (2002) for more details on the treatment of other null elements by the translation algorithm.

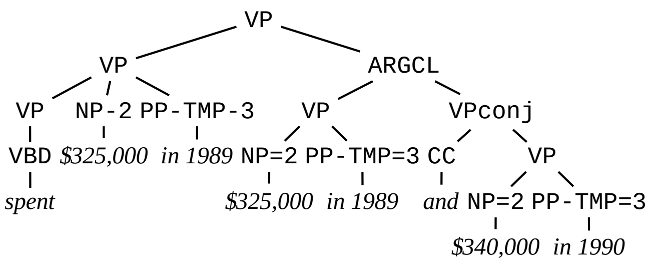
Argument-cluster coordination The Treebank assumes the following bracketing for argument-cluster coordination:

```
(S (NP-SBJ (PRP they))
   (VP (VP (VBD spent)
           (NP-2 $325,000)
           (PP-TMP-3 in 1989))
       (CC and)
       (VP (NP=2 $340,000)
           (PP-TMP=3 in 1990))))
```

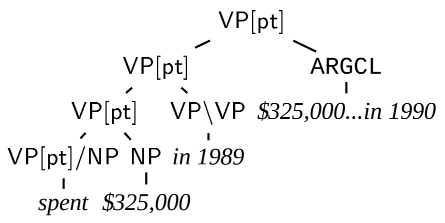
The "=" notation (also called gap coindexing) expresses that the arguments in the second conjunct have the same roles as their counterparts in the first conjunct. The Treebank tree does not correspond to the CCG analysis of such constructions (using VP to abbreviate S\NP):



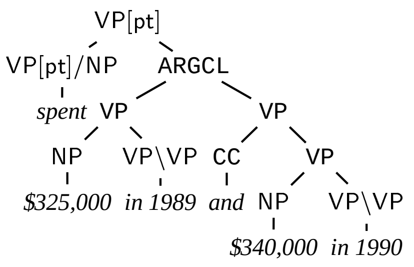
We obtain this tree by creating a new node ARGCL consisting of a VP consisting of copies of the co-indexed elements of the first conjunct, the conjunction and the second conjunct (again using VP to abbreviate S\NP):



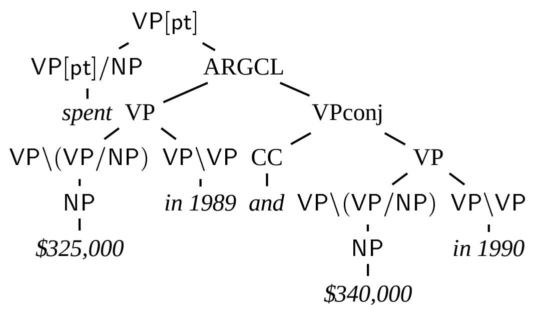
Category assignment now proceeds in two phases: first we assign categories in the normal fashion, ignoring the ARGCL tree:



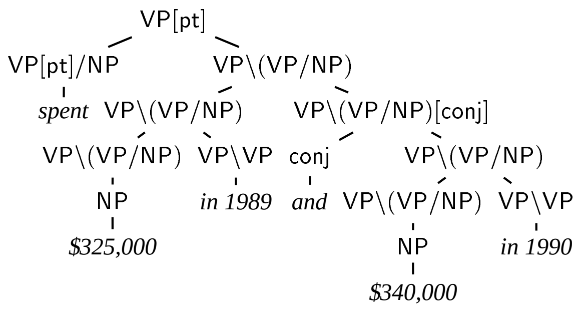
Then the constituents which are coindexed with the elements in the first tree are assigned the same categories as their antecedents, and all nodes in the first conjunct apart from the verb are cut out:



Then category assignment proceeds underneath the coindexed nodes (not shown here), as well as above them. Category assignment within an argument cluster is a bottom-up, right-to-left process. The leftmost node (PP-TMP=3) is an adjunct and does not need to be type-raised. However, the object noun phrase is backward-typeraised to VP\(VP/NP) (instantiating the T in the type-raising rule with the category of the parent of the argument cluster). If there was another object to the left of this noun phrase, with category Y, its type-raised category would be (VP/NP)\((VP/NP)/Y), instantiating the T with the argument category of the previous complement.



Then category assignment proceeds, bottom-up:



Here is the complete translation algorithm, including the preprocessing step described in section 5.:

```
foreach tree $\tau$:
        preprocessTree($\tau$);
        preprocessArgumentClusters($\tau$);
        determineConstituentType($\tau$);
        makeBinary($\tau$);
        decorateTree($\tau$);
        assignCategories($\tau$);
        treatArgumentClusters($\tau$);
        cutTracesAndUnaryRules($\tau$);
```

Future work As it stands, the translation algorithm cannot deal with sentences involving gapping, handled by a decomposition rule in CCG. Furthermore, the coordination of constituents with unlike categories is not possible at present. However, 98.3% of the sentences of sections 02-21 of the Treebank can be processed by this algorithm.

5. Preprocessing the Treebank

There are certain well-known problems with the Penn Treebank markup – for instance, the noun phrase structure is very flat; the complement-adjunct distinction is not always marked up in a very consistent manner, and there is an estimated POS-tagging error rate of 3% (Ratnaparkhi, 1996). The translation algorithm is sensitive to these errors: POS-tagging errors can lead to incorrect categories or to incorrect features on verbal categories (eg. when a past participle is wrongly tagged as past tense); the omission or addition of functional tags causes errors in the complement-adjunct distinction; and certain types of coordinate constructions are not recognized as such if the bracketing is not correct. Additionally, the algorithm presented in section 4 requires the constituent structure of the phrase-structure tree before binarization to conform to the desired CCG analysis. For instance, if the flat tree of a coordinate construction contains any adjuncts or arguments to the conjuncts, a separate level has to be inserted before binarization can proceed. This is true for constituents which are co-indexed with a right-node-raising trace (*RNR*), but there are also other cases which are either not explicitly marked as right-node-raising constructions, or where adjuncts to one of the conjuncts appear as sisters rather than daughters of the consituent they modify.

Cleaning up the Treebank We attempt to correct POS-tag errors that are likely to lead to errors in the translation process. For instance if a simple past tense form occurs in a verb phrase which itself is the daughter of a verb phrase whose head is an inflected verb, it is highly likely that it should be a past participle instead. Using the verb form itself and the surrounding context, we attempt to correct such errors automatically.

Alterations to Treebank analyses In order to obtain the desired categorial analyses, a few alterations to the original Treebank analyses are made. The most important of these is the insertion of a noun level into the otherwise flat NP structure in the Treebank. We also employ various heuristics to impose a structure on flat quantifier phrases (QP) and possessives. As mentioned above, a separate level consisting only of the conjuncts is introduced in coordinate constructions, and modifiers of conjuncts are made into daughters of these. Furthermore, we adopt a different analysis of small clauses, which necessitates further changes to the Treebank (see (Hockenmaier et al., 2002) for details).

Remaining problems with the Treebank A number of obstacles for the translation to a linguistically richer formalism such as CCG remain. The flat noun phrase structure is one of them: although it is possible to introduce a separate noun level, compound nouns still have a flat internal structure, which is semantically undesirable. The Treebank markup also makes appositives difficult to distinguish from noun phrase lists. Moreover, postnominal modifiers are always attached at the NP-level which is also undesirable from a semantic point of view.

Within the verb phrase and sentential structure, there are other, equally well known problems – therefore it is standard for Parseval scores reported for parsers trained on the Penn Treebank to conflate the labels ADVP and PRT. The distinction between complements and adjuncts is similarly difficult to draw, especially for constituents annotated with the CLR-tag, which is known to be used fairly inconsistently across the corpus.

6. Type-changing rules

The basic algorithm described in section 4 leads to a proliferation of adjunct categories, as illustrated in figure 1. For example, a past participle such as *used* receives different categories depending on whether it occurs in a reduced relative or a main verb phrase. As a consequence, modifiers of *used* will also receive different categories depending on what occurrence of *used* they modify. This is clearly undesirable, since we are only guaranteed to acquire a complete lexicon if we have seen participles (and all their possible modifiers) in all their possible surface positions. Similar regularities have been recognised and given a categorial analysis by Carpenter (1992), who advocates lexical rules to account for the use of predicatives as adjuncts. Carpenter also proposes lexical rules for passivization, as well as for the predicative use of noun phrases, prepositional phrases and adjectives (which he analyzes as S[pred]\NP) and the use of auxiliaries in yes-no questions. All of these constructions are analyzed in our corpus with separate lexical entries.

However, the use of predicatives as adjuncts leads to a proliferation of categories, not only for the predicatives themselves, but also for their modifiers. Carpenter captures this by a lexical rule, but it seems more economical to put such type-changing rules in the grammar. Such an approach has been taken by Aone and Wittenburg (1990) (also in a categorial framework) to encode morphological rules, but also for reduced relatives and other syntactic constructions. Aone and Wittenburg show that these type-changing rules can be derived from zero morphemes in the grammar. Carpenter (1991) and (1992) show that, in general, the inclusion of lexical rules can lead to a shift in generative power from context-free to recursively enumerable. However, this does not hold true if these lexical rules cannot operate on their own output and hence generate an infinite set of category types. Like Aone and Wittenburg, we only consider a finite number of instantiations of these type-changing rules, namely those which arise when we extend the category assignment procedure in the following way: for any sentential or verb phrase modifier (an adjunct with label S,SBAR with null complementizer, or VP) to which the original algorithm assigns category X|X, apply the following type-changing rule in reverse:

(5) $\mathsf{S\$} \Rightarrow \mathsf{X|X}$

where S$ is the category that this constituent obtains if it is treated like a head node by the basic algorithm. S$ has the appropriate verbal features, and can be S\NP or S/NP. Some of the most common type-changing rules are:

(6) $\mathsf{S[pss]\backslash NP} \Rightarrow \mathsf{NP\backslash NP}$
"workers [exposed to it]"

(7) $\mathsf{S[adj]\backslash NP} \Rightarrow \mathsf{NP\backslash NP}$
"a forum [likely to bring attention to the problem]"

(8) $\mathsf{S[ng]\backslash NP} \Rightarrow \mathsf{NP\backslash NP}$
"signboards [advertising imported cigarettes]"

(9) $\mathsf{S[ng]\backslash NP} \Rightarrow \mathsf{(S\backslash NP)\backslash(S\backslash NP)}$
"become chairman, [succeeding Ian Butler]"

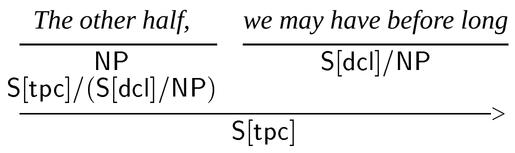
(10) $\mathsf{S[dcl]/NP} \Rightarrow \mathsf{NP\backslash NP}$
"the millions of dollars [it generates]"

The other main unary type-changing rule that we make use of in the grammar is a rule which allows noun phrases (NP) to be derived from nouns (N). Carpenter proposes a similar lexical rule.

6.1. Topicalization

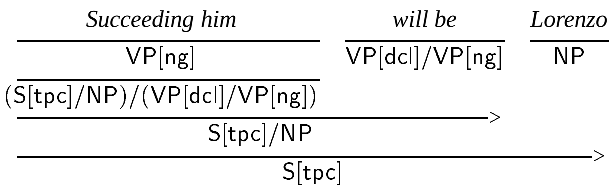
Following (Steedman, 1987), we assume that English has the following schema of a non order-preserving type-raising rule to account for topicalization of noun phrases, adjective phrases and prepositional phrases:

(11) $\mathsf{X} \Rightarrow \mathsf{S[topic]/(S[dcl]\backslash X)}$
with $\mathsf{X} \in \{\mathsf{NP}, \mathsf{PP}, \mathsf{S[adj]\backslash NP}\}$



Similarly, we use the following unary type-changing rule scheme for verb phrase topicalization (with $[x] \in \{[\mathsf{ng}], [\mathsf{pss}]\}$):

(12) $\mathsf{VP[x]} \Rightarrow \mathsf{(S[tpc]/NP)/(VP[dcl]/VP[x])}$



6.2. Binary type-changing rules

In written English, certain types of NP-extraposition require a comma before or after the extraposed noun phrase:

(13) *Factories booked \$236.74 billion in orders in September, [$_{NP}$ nearly the same as the \$236.79 billion in August]*

We make use of this fact in the following binary type-changing rules:

NP	,	⇒	S/S
,	NP	⇒	S\S
,	NP	⇒	(S\NP)\(S\NP)

7. The effect of preprocessing and type-changing rules

How much does preprocessing change the treebank? We can compare the modified treebank (before binarization and translation to CCG) with the original treebank, using standard Parseval scores. On section 00, the modified treebank has labelled precision of 99.1% and labelled recall of 74.65% (or 97.02% if noun labels, which are also introduced in the preprocessing step are ignored in the evaluation). The POS-tag accuracy is 99.5%. However, a complete match between the modified and original Treebank is only obtained for 0.47% of the sentences, or 60.12% if noun labels are ignored. Therefore, the effect on the obtained CCG corpus is much bigger than the precision and recall figures suggest.

Table 1 shows how the size of the grammar and lexicon extracted from CCGbank (sections 02-21) changes if preprocessing and type-changing rules (TCR) are introduced. Preprocessing reduces the size of the lexicon and grammar by 4.6% (cf. rows 1 and 2), whereas the main effect of the

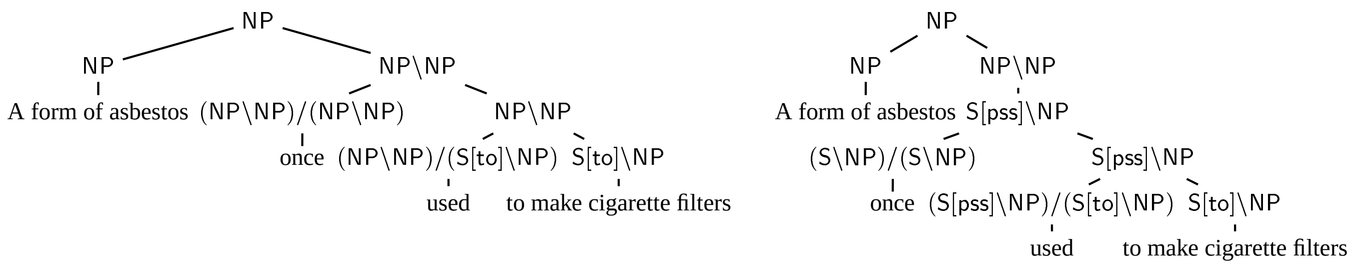


Figure 1: The effect of type-changing rules on lexical categories of adjuncts

type-changing rules (row 3) is to reduce the number of category types. In total, the average number of categories per words decreases by 9%. This grammar is also much smaller (and less overgenerating) than a phrase structure grammar extracted from the original Penn Treebank, which has more than 12,000 rules.

This reduction is important, because one of the biggest problems for expressive grammar formalisms is lexical coverage on unseen text. Table 1 also shows the coverage of a lexicon extracted from sections 02-21 on section 00. Preprocessing and type-changing rules reduce the problem of encountering unseen pairings of seen words and seen categories (unseen ⟨sw,sc⟩) by more than 26% from 3.0% to 2.2%, simply by reducing the category inventory.

When a corpus is used to train statistical models such as parsers or taggers, it is common to replace rare words in the training data with a token UNKNOWN, and to replace unseen words in the test data with this token. A similar approach can be used for CCG and other lexicalized grammar formalisms, but because of the large number of categories, it is better to adopt a more fine-grained analysis of rare and unknown words. Using the POS-tags in the treebank, we can distinguish rare and unknown words with different parts of speech – instead of having one token UNKNOWN which stands for all unknown words, we acquire lexical entries for UNKNOWN-NNS, UNKNOWN-VBZ etc. When using this lexicon during parsing, it has to be assumed that the input is POS-tagged. Table 2 shows how lexical coverage increases if words with a frequency of 2 or less in the training data and unknown words in the test data are replaced with their POS-tag, and if all digits are replaced with a token "X".

Row 1 shows the impact this treatment of unknown words has on the lexicon without preprocessing or type-changing rules. This compares with the figures in row 2, which show that preprocessing and type-changing still have an important effect. The problem of unseen word-category pairings for seen words and category types is reduced by over 35% in comparison to row 1, and by 50% in comparison to the baseline figures reported in table 1. Thus, the combination of preprocessing, type-changing rules and an adequate treatment of unknown words results in a lexicon with a coverage of 98.5% of the tokens in unseen data, as compared with a baseline of 93.2%.

8. Conclusion

We have presented an algorithm which translates the Penn Treebank into a corpus of CCG normal-form derivations, and have outlined the kinds of changes that need to be performed on the trees before this algorithm can applied in order to obtain correct CCG analyses. As a byproduct we have also produced a cleaner version of the original treebank, which we hope to make available to the NLP community in a generally useable form in due course. Furthermore, we have shown that the introduction of a small number of type-changing rules in the grammar leads to a reduction in the size of the acquired lexicon. Together with the required preprocessing this alleviates the lexical coverage problem engendered by CCG's rich category set.

We plan to extend this algorithm to also deal with gapping and unlike-coordinate phrases (UCP), in order to obtain complete coverage of the original Treebank. We also plan to investigate how an off-line application of rules of inflectional and derivational morphology to the acquired lexicon will affect its size and coverage.

Since every CCG derivation corresponds to the construction of a semantic interpretation, the work presented in this paper is a first step towards the creation of a treebank annotated with semantic interpretations.

9. Acknowledgements

The research was funded in part by EPSRC grant GR/M96889, by an EPSRC studentship, and by the Edinburgh Language Technology Group. We would like to thank Stephen Clark, Jason Baldridge and Miles Osborne for many helpful discussions and comments.

10. References

C. Aone and K. Wittenburg. 1990. Zero morphemes in Unification-Based Combinatory Categorial Grammar. In *28th Annual Meeting of the Association for Computational Linguistics*, pages 188–193, University of Pittsburgh, Pittsburgh.

B. Carpenter. 1991. The Generative Power of Categorial Grammars and Head-Driven Phrase Structure Grammars with Lexical Rules. *Computational Linguistics*, 17(3):301–314.

B. Carpenter. 1992. Categorial Grammars, Lexical Rules, and the English Predicative. In Robert Levine, editor, *Formal Grammar: Theory and Implementation*, chapter 3. Oxford University Press.

J. Chen and K. Vijay-Shanker. 2000. Automated Extraction of TAGs from the Penn Treebank. In *Proceedings of the 6th International Workshop on Parsing Technologies*, Trento, Italy.

S. Clark, J. Hockenmaier, and M. Steedman. to appear. Building Deep Dependency Structures Using a Wide-coverage CCG Parser. In *Proceedings of the 40th Annual*

		Grammar and lexicon size				Lexical coverage			
						seen	unseen	unseen	unseen
Preprocessing	Type-changing	#rules	#cats	# entries	cats/word	⟨sw, sc⟩	⟨sw, sc⟩	⟨sw, uc⟩	⟨uw, uc⟩
No	No	4054	1301	80,597	1.83	93.2%	3.0%	0.05%	3.76%
Yes	No	3239	1234	76,247	1.74	93.7%	2.4%	0.01%	3.83%
No	Yes	3871	1206	77,973	1.78	93.4%	2.7%	0.05%	3.76%
Yes	Yes	3086	1129	73,739	1.68	94.0%	2.2%	0.02%	3.83%

Table 1: Size of grammar and lexicon extracted from sections 02-21, and lexical coverage on section 00

Preprocessing	Type-changing	seen ⟨sw, sc⟩	unseen ⟨sw, sc⟩	unseen ⟨sw, uc⟩	unseen ⟨uw, uc⟩	#cats/word
No	No	97.6%	2.3%	0.05%	0%	3.06
Yes	Yes	98.5%	1.5%	0.02%	0&	2.65

Table 2: Impact of treating unknown words on lexical coverage

Meeting of the Association for Computational Linguistics.

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

J. Eisner. 1996. Efficient Normal-Form Parsing for Combinatory Categorial Grammar. In *Proceedings of 34th Annual Meeting of the Association for Computational Linguistics, Santa Cruz, June*.

A. Frank, L. Sadler, J. van Genabith, and A. Way. Forthcoming. From Treebank Resources to LFG F-structures. In Anne Abeillé, editor, *Treebanks. Building and using syntactically annotated corpora*. Kluwer Academic Publishers.

G. Gazdar, E. Klein, G.K. Pullum, and I.A. Sag. 1985. *Generalised Phrase Structure Grammar*. Blackwell, Oxford.

M. Hepple and G. Morrill. 1989. Parsing and Derivational Equivalence. In *Proceedings of the 4th Conference of the European Association for Computational Linguistics, Manchester*, pages 10–18, April.

J. Hockenmaier and M. Steedman. to appear. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

J. Hockenmaier, G. Bierner, and J. Baldridge. to appear. Extending the Coverage of a CCG System. In *Journal of Logic and Computation*.

L. Karttunen. 1989. Radical Lexicalism. In M.R. Baltin and A.S. Kroch, editors, *Alternative Conceptions of Phrase Structure*. Chicago University Press, Chicago.

E. König. 1994. A Hypothetical Reasoning Algorithm for Linguistic Analysis. *Journal of Logic and Computation*, 4:1–19.

D.M. Magerman. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford.

A. Ratnaparkhi. 1996. A Maximum Entropy Part-of-speech Tagger. In *Proceedings of the EMNLP Conference*, pages 133–142, Philadelphia, PA.

M. Steedman. 1987. Combinatory Grammars and Parasitic Gaps. *Natural Language and Linguistic Theory*, 5:403–439.

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

F. Xia. 1999. Extracting Tree Adjoining Grammars from Bracketed Corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium(NLPRS-99)*.