Translating Treebank Annotation for Evaluation

Stephen Watkinson and Suresh Manandhar

Department of Computer Science,
University of York,
York YO10 5DD,
UK.
stephen.watkinson@cs.york.ac.uk
suresh.manandhar@cs.york.ac.uk

Abstract

In this paper we discuss the need for corpora with a variety of annotations to provide suitable resources to evaluate different Natural Language Processing systems and to compare them. A supervised machine learning technique is presented for translating corpora between syntactic formalisms and is applied to the task of translating the Penn Treebank annotation into a Categorial Grammar annotation. It is compared with a current alternative approach and results indicate annotation of broader coverage using a more compact grammar.

1 Introduction

Annotated corpora have become a vital tool for Natural Language Processing (NLP) systems, as they provide both a standard against which results can be evaluated and a resource from which to extract linguistic information e.g. lexicons. This is especially true in any NLP task that requires the annotation of examples, e.g. part-of-speech tagging, parsing and semantic annotation, where it is vital to have a correct standard against which to compare the results of systems attempting to solve the task. Similarly, it is crucial in a language learning context, where what is learned can be used to annotate examples e.g. syntax learning, lexical learning. In this case the learned artefact is used to annotate the examples, which can then be compared against the correctly annotated version. Hence, correctly annotated corpora are vital for the evaluation of a very large number of NLP tasks.

Unfortunately, there are often no suitably annotated corpora for a given task. For example, the Penn Treebank (Marcus et al., 1993; Marcus et al., 1994; Bies et al., 1994) provides a large corpus of syntactically annotated examples mostly from the Wall Street Journal. It is an excellent resource for tasks dealing with the syntax of written English. However, if the annotation formalism (a phrase-structure grammar with some simple features) does not match that of one's NLP system, it is of very little use. For example, suppose a parser using Categorial Grammar (Wood, 1993; Steedman, 1993) is developed and applied to the examples in the corpus. While the bracketing of the examples will bear a strong relationship to the bracketing of the treebank, the labelling of the lexical items and the inner nodes of the tree will be entirely different and no labelling evaluation will be possible.

However, intuitively, plenty of syntactic information is available. In fact, for most evaluation, all the syntactic information required is available, but in the wrong form. It seems obvious that a system for translating the syntactic information between formalisms would be a useful tool.

Here, we present a system that translates the annotation of the Penn Treebank from the standard phrase structure annotation to a Categorial Grammar (CG) annotation and in the process induces large scale CG lexicons. It is a data-driven multi-pass system that uses both predefined rules

and machine learning techniques to translate the trees and in the process induce a large scale CG lexicon. The system was designed to produce the lexical annotations for the sentences without null elements (i.e. without movement) from the Penn Treebank, so that these could be used to evaluate the results produced by an unsupervised CG lexicon learner (Watkinson and Manandhar, 2000; Watkinson and Manandhar, 2001).

The system has four major features. Firstly, there is significant control over how the treebank is annotated. This is vital if the results are to be used for evaluation. Secondly, the system prevents propagation of translation errors throughout the trees by being data-driven. Thirdly, the system deals elegantly with erroneous annotation, even providing a degree of self-correction. Finally, the approach is general enough to apply to other similar problems.

The system is compared with a top-down alternative based on the algorithm of Hockenmaier *et al* (Hockenmaier et al., 2000), which is currently the system which has been applied to the most similar task, although it is really for CG lexicon extraction. The comparison suggests that the algorithm presented here gives more compact and linguistically elegant solutions. Investigation also indicates that the corpus produced is effectively translated for its purpose.

In Section 2 other work in the area is briefly reviewed. In Section 3 the precise translation task is described. This is followed in Section 4 with a detailed description of the algorithms used for this task and some discussion as to their appropriateness. The results from the experiments are in Section 5. Finally, in Section 6 the results are discussed along with the contributions of the work and some suggestions for future work.

2 Previous Work

The most appropriate work to consider within this context is the grammar extraction literature. Perhaps the earliest example is the approach of Charniak (Charniak, 1996), who simply extracted a context-free grammar by reading off the production rules implied by the trees in the Penn Treebank. While not translating the formalism of the treebank, this has led to work extracting grammars of different formalisms.

The majority of work is based on the most obvious extension of the Charniak approach, which is to extract subtree-based grammars e.g. the Data-Oriented Parsing (DOP) approach (Bod, 1995), or extracting Lexicalised Tree Adjoining Grammars (LTAGs), or more generally Lexicalised Tree Grammars (LTGs) (Neumann, 1998; Xia, 1999; Chen and Vijay-Shanker, 2000). Each approach involves a process that splits up the annotated trees in the treebank into a set of subtrees that define the grammar. These approaches still continue to work with the syntactic data in the same form as it is found in the corpora.

A slightly different approach has been followed by Krotov *et al* (Krotov et al., 1998), where they extract the grammar from the Penn Treebank like Charniak, but then compact it. This provides a smaller grammar of similar quality to a grammar that has not been compacted, when a linguistically motivated compaction is used. However, the formalism remains unchanged. Similarly, Johnson (Johnson, 1998) modifies the labelling of the Penn Treebank, but remains within a CFG framework.

Hockenmaier *et al* (Hockenmaier et al., 2000), although to some extent following the approach of Xia (Xia, 1999) where LTAGs are extracted, have pursued an alternative by extracting Combinatory Categorial Grammar (CCG) (Steedman, 1993; Wood, 1993) lexicons from the Penn Treebank. In this case the data in the treebank is truly translated into another formalism providing an entire CCG annotation for the corpus based on a top-down algorithm. The lexicon is built by reading off the lexical assignments made for each tree. This is the most closely related work to this research, especially as it translates into a formalism very closely related to CG.

The algorithm presented by Hockenmaier *et al* (Hockenmaier et al., 2000) has been used to build a top-down system against which to compare our data-driven system. The algorithms are both described in detail in Section 4.

3 The Task

Given a subset of the examples from the Penn Treebank annotated with syntactic and part-of-speech information (slightly modified), the system should return the examples annotated with

the correct CG categories attached to the words of the sentence and the lexicons these imply.

The context of the task explains some parts of its definition. The translated corpus is to be used as a standard against which to compare the lexical annotation (i.e. the categories assigned to the words) of the output of an unsupervised CG learner that annotated the words of the examples with CG categories and then extracts a probabilistic lexicon (see Watkinson and Manandhar (Watkinson and Manandhar, 2001) for details). Hence, there is no need for specific tree annotation. The learner currently uses a slightly modified subset of the treebank, which is described below.

3.1 The Corpus

The systems are applied to examples from the Penn Treebank (Marcus et al., 1993; Marcus et al., 1994; Bies et al., 1994) a corpus of over 4.5 million words of American English annotated with both part-of-speech and syntactic tree information.

To be exact, we are using the Treebank II version (Bies et al., 1994; Marcus et al., 1994), which attempts to address the problem of complement/adjunct distinction, which previous versions had ignored. While the documentation is clear that the complement/adjunct structure is not explicitly marked (Marcus et al., 1994), the annotation includes a set of labels that relate to the role of a particular constituent in the sentence. These labels are attached to the standard constituent label and it is possible to use heuristics to determine the probable complement/adjunct structure in the trees (Collins, 1999; Xia, 1999), which is obviously useful in translating the annotation.

The full Penn Treebank is not being used. As mentioned already, the current research only uses sentences without null elements (i.e. without movement) from the treebank and does not include any of the sentence fragments. However, as Categorial Grammar formalisms do not usually change the lexical entries of words to deal with movement, but use further rules (Wood, 1993; Steedman, 1993; Hockenmaier et al., 2000), the lexicons learned here will be valid over corpora with movement. The extracted corpus, C1, in fact contains 5000 of the declarative sentences of fifteen words or less (although the sentence length makes little difference to either of the translation procedures described) from the Wall Street Journal section of the treebank. To give an indication of the complexity of the corpus, the number of tokens, i.e. the total number of words including repetitions of the same word, is 47,782. The total number of unique words, i.e. not including repetitions of the same word, is 12,277. We also extracted C2, a 1000 example corpus (also of declarative sentences from the Wall Street Journal section) with 9467 tokens and 3731 words, which is used in the evaluation process.

The corpora also have some small modifications, which mean that adjacent nominals in the same subtree are combined to form a single nominal and the punctuation is removed. These modifications are made for use with the unsupervised learner (Watkinson and Manandhar, 2000; Watkinson and Manandhar, 2001) to simplify the learning process. They may also slightly simplify the translation process, but it is necessary for the corpus annotation that we want.

3.2 Categorial Grammar

Categorial Grammar (CG) (Wood, 1993; Steedman, 1993) provides a functional approach to lexicalised grammar, and so can be thought of as defining a syntactic *calculus*. Below we describe the basic (AB) CG. The current work uses this simple form of the grammar, which suffices for the syntactic annotation of the corpora currently being used.

There is a set of *atomic* categories in CG, which are usually nouns (n), noun phrases (np) sentences (s) and sometimes prepositional phrases (pp), although this can be consider shorthand for the full category (Wood, 1993). It is then possible to build up *complex* categories using the two slash operators "/" and "\". If A and B are categories then A/B and A\B are categories, where (following Steedman's notation (Steedman, 1993)) A is the resulting category when B, the argument category, is found. The direction of the "slash" functors indicates the position of the argument in the sentence *i.e.* a "/" indicates that a word or phrase with the category of the argument should immediately *follow* in the sentence. With the "\" the word or phrase with the argument category should

immediately *precede* the word or phrase with this category. This is most easily seen with examples.

Suppose we consider an intransitive verb like "run". The category that is required to complete the sentence is a subject noun phrase. Hence, the category of "run" is a sentence that is missing a preceding noun phrase *i.e.* s\np. Similarly, with a transitive verb like "ate", the verb requires a subject noun phrase. However, it also requires an object noun phrase, which is attached first. The category for "ate" is therefore (s\np)/np.

With basic CG there are just two rules for combining categories: the forward (FA) and backward (BA) *functional application* rules. Following Steedman's notation (Steedman, 1993) these are:

$$X/Y\ Y \Rightarrow X \qquad (FA)$$
$$Y\ X\backslash Y \Rightarrow X \qquad (BA)$$

where X and Y are CG categories. In Figure 1 the parse derivation for "John ate the apple" is presented, showing examples of how these rules are applied to categories.

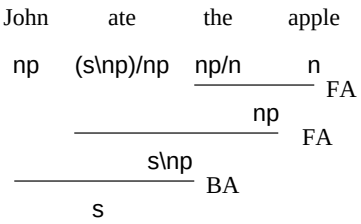

Figure 1: An Example Parse in Basic CG

The CG formalism described above has been shown to be weakly equivalent to context-free phrase structure grammars (Bar-Hillel et al., 1964). While such expressive power covers a large amount of natural language structure, it has been suggested that a more flexible and expressive formalism may capture natural language more accurately (Wood, 1993; Steedman, 1993). In future we may consider applying the principle developed here to perform translations to these more complex formalisms, although many of the changes will not actually change the lexical entries, just the way they can be combined.

4 Alternative Approaches

This section presents the two approaches to translation that are being compared. Firstly, there is

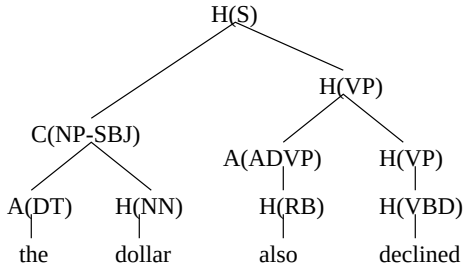

Figure 2: A tree with constituents marked

the top-down method, which is a version of the algorithm described by Hockenmaier *et al* (Hockenmaier et al., 2000), but used for translating into simple (AB) CG rather than the Steedman's Combinatory Categorial Grammar (CCG) (Steedman, 1993). The algorithm here does not need to deal with movement, as the corpus does not contain any. The atomic pp category is included in the CG with this approach, but not with our approach, as it is a convenient shorthand for the prepositional phrase category.

The second approach is a multiple-pass data-driven system. Rules for translating the trees are applied in order of complexity starting with simple part-of-speech translation and finishing with a category generation stage.

4.1 Top-Down Category Generation

The algorithm has two stages.

Mark constituents All the nodes of all trees are marked with their roles i.e. as heads, complements or adjuncts. While Hockenmaier *et al* (Hockenmaier et al., 2000) are unclear, it is assumed that this is achieved using heuristics. Collins (Collins, 1999) describes such a set of heuristics, which are used with some minor modifications for CG and the changed Penn Treebank annotation. Figure 2 shows an example of an annotated tree.

Assign categories This is a recursive top-down process, where the top category in the tree is an s. The category of the complements is determined by a mapping between Treebank labels and categories e.g. NP in the treebank becomes np. Hockenmaier *et al* (Hockenmaier et al., 2000) do not provide the mapping, so it was built specially for this system. This mapping led to the inclusion

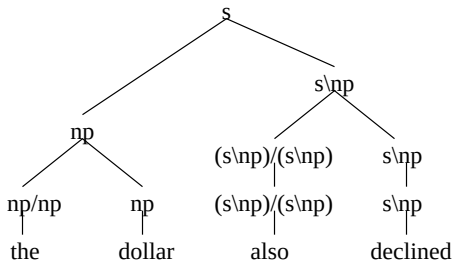

Figure 3: An example with categories assigned

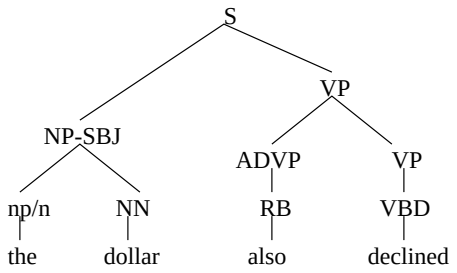

Figure 4: Example of the output of Stage 1

of the pp category as shorthand for prepositional complements. It should make no difference to the annotation process, but could lead to the generation of a few more categories. The head child of a subtree is given the category of the parent plus the complements required, which are found by looking first to the left of the head and then to the right, and adding them in the order they should processed in. Finally, adjuncts are assigned the generic X/X or $X\backslash X$ where X is the head category with the complements removed which have been dealt with before the adjunct is processed. Figure 3 shows an example of a tree with the categories assigned to it.

This algorithm has several advantages. It is simple and robust and has been shown by Hockenmaier *et al* (Hockenmaier et al., 2000) to provide good lexical annotation leading to useful CCG lexicons.

However, it has two main disadvantages. Firstly, there is no control over category generation other than the rather weak constraints of the formalism and the heuristic syntactic roles. This is likely to lead to some linguistically implausible annotation. Secondly, the top-down nature of the algorithm is likely to lead to any translation errors being propagated down the tree, which will lead to some unusual and large categories, as Hockenmaier *et al* (Hockenmaier et al., 2000) report.

4.2 Bottom-Up Sequential

Our system uses a four stage process, where the type of translation changes at each stage.

4.2.1 Stage 1: Parts-of-Speech

This is the simplest level of translation. The mapping between the Penn Treebank part-of-speech annotation and the CG category annotation is many-to-many, but some parts-of-speech can be translated directly into categories using simple rules e.g. the following rule states that words with the determiner part of speech (DT) can be translated into the CG category np/n.

$$\mathrm{DT} \rightarrow \mathsf{np/n}$$

The system passes through the full set of examples and translates the appropriate parts-of-speech. See Figure 4 for an example of the output of this stage.

4.2.2 Stage 2: Subtrees

The next pass through the data allows more complex rules to be used. Consider the part-of-speech label NNS, used in the Penn Treebank annotation scheme to indicate a plural noun. Its syntactic role can be that of a simple noun (n) or a noun phrase (np), so we need a mechanism for choosing between these two possibilities.

The most obvious mechanism is to use the surrounding subtree to provide the context to select the correct rule. If the NNS tag is part of a noun phrase which begins with something fulfilling the determiner role, then the tag should be translated to the CG category n, otherwise it should be translated as an np.

The algorithm for applying the set of context-based rules is a simple matching process throughout the treebank. Figure 5 shows the output from this stage on an example.

4.2.3 Stage 3: Structural Heuristic

In this stage, the system uses further knowledge to attempt to inform the translation process. Where words have not been translated, the system annotates the subtree with the head, complements and adjuncts using a modified version of Collins' heuristics (Collins, 1999).

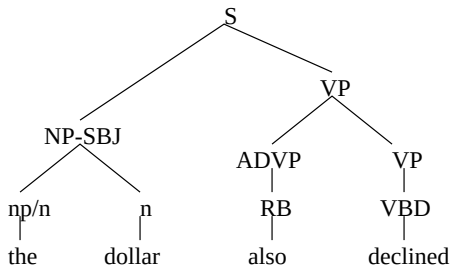

Figure 5: Example of the output of Stage 2

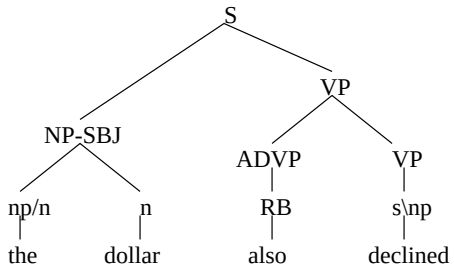

Figure 6: Example of the output of Stage 3

Further categories can now be obtained. For example, if the head of the subtree requires an np category to its right as its first complement and there is a word marked as a complement in this position, then it can be translated as an np. Alternatively, if the head category is unknown, but it is verbal according to the Penn Treebank label then looking at the categories of the complements can determine the type of verb it is e.g. no complements following a verb indicates a CG category s\np. Figure 6 shows the effects of this stage on the example.

4.2.4 Stage 4: Category Generation

In the final stage each lexical category that has not been annotated is given a variable for a category. The tree is then traversed bottom-up instantiating these categories by using head, complement and adjunct annotation and the already annotated categories. The building of head and adjunct categories follows the same process described for the top-down algorithm. Complements either gain their categories through this process or have already had them assigned. Figure 7 shows the final output.

This approach has two main advantages. Firstly, the user has control over the type of CG to which the treebank is translated, due to the

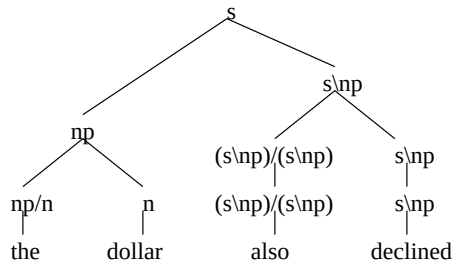

Figure 7: Example of the output of Stage 4

use of predefined categories for predefined contexts. Secondly, the bottom-up approach ensures that translation errors are not propagated seriously through the tree.

A further advantage exists that has not, as yet, been fully investigated. The system, due to its multi-pass nature, has the potential for translations to clash. Experience has shown that this occurs when there is an annotation error, so the system can be used to highlight these and can also provide some level of self-correction. This has not been investigated in detail, but the current approach, which gives satisfactory results, is to assume the head category is correct and adjust complements and adjuncts accordingly. In future, a simple correction scheme could easily be added to produce a self-correcting translator.

The main weakness of the system is the reliance upon the head/complement/adjunct annotating heuristics, which were not designed to be used with a CG.

The system also returns some categories with variables. This is due in part to the heuristics and in part to the small number of rules currently used in the early stages of the translation process. Most of the problem categories could be dealt with by the addition of a few more rules in stages 2 and 3.

5 Results

Here we provide similar evaluation of the systems as others (Hockenmaier et al., 2000; Xia, 1999) for easy comparison. Both systems were used translate C1 and C2. C2 is used for determining the coverage of the grammar used by the two systems. Both systems, at times, failed to translate examples (frequently due to annotation error in the original treebank). The top-down system failed on 60 and 15 examples from C1 and C2

	Top-down	Bottom-up
No. of cats	167	106
Lexicon size	15887	15136
Ave. cats/word	1.31	1.25
Ave. cat size	8.02	5.12

Table 1: Table of category and lexicon information on the translated corpora

Freq. Range	Number of Categories	
	Top-down	Bottom-up
1≤f≤1	42	29
2≤f≤10	61	34
11≤f≤20	14	9
21≤f≤100	24	11
101≤f≤1000	17	13
1001≤f≤5000	7	7
5001≤f≤10000	1	2
10001≤f≤12000	0	1
12001≤f≤15000	1	0

Table 2: Table of the category frequencies for both approaches

respectively. The bottom-up system failed on 66 and 15 examples from C1 and C2 respectively.

Table 1 describes the type of categories used to translate C1 and the size of the lexicons generated. Categories with variables in were ignored, as they could usually be unified with an already existing category. With this in mind, the bottom-up algorithm extracted a more compact lexicon. The average category sizes (the number of slash operators in categories) are interesting, as they indicate the profligacy of the top-down algorithm in creating unwieldy categories, whereas the bottom-up approach uses smaller and, on inspection, more plausible categories. These results seem, in part, to vindicate the choice of a controlled bottom-up approach.

Tables 2 and 3 present the results for both systems for the frequency distribution of categories (i.e. the number of categories that appeared with a particular frequency) and the frequency distribution of the number of categories for a word (i.e. the number of words that had a particular number of categories). The trends for both systems are similar. There are a large number of categories that appear very infrequently, these tend

Freq. Range	Word frequency	
	Top-down	Bottom-up
f=1	10486	10377
f=2	1263	1264
f=3	264	264
f=4	86	86
$5 \leq f \leq 9$	100	100
$10 \leq f \leq 14$	20	20
$15 \leq f \leq 24$	10	10
$25 \leq f \leq 30$	2	2

Table 3: Frequencies of words appearing in a frequency range of number of categories

to be the larger, generated categories and often fit unusual circumstances e.g. misannotation of the treebank, or mistakes in the use of the heuristics. The bottom-up approach has many fewer of these categories, indicating the problem of propagating of errors down the tree with the top-down approach. There are also a few exceptionally frequent categories, these are noun phrases, nouns, and some of the common verbs.

The number of categories per word is similar, suggesting the approaches are similar in their ability to produce the variety of categories required for words.

While these figures give some indication of the quality and compactness of the translation, it is useful to determine the coverage of the lexicon extracted from C1 by comparing it with a lexicon extracted from C2 and so determine the quality and generality of the lexicon that has been produced in the translation. Table 4 shows the comparison. Here *entry* means the C1 lexicon contains an entry the same as the C2 entry. *kwkc* means that the entry from C2 is not in C1, but both the word and the category are known. *kwuc* means the word is in the C1 lexicon, but the category is not. Finally, *uw* indicates that the word is in C1. Despite a smaller lexicon and a smaller number of categories, the bottom-up system gives better coverage. Note especially that there are no unknown categories with with the bottom-up approach and that the percentage of exact entries is much higher.

	Top-down	Bottom-up
Categories	98	65
New categories	4	0
entry %	37.29	48.31
kwkc %	10.55	11.09
kwuc %	11.46	0
uw %	40.70	40.60

Table 4: Table comparing the coverage of the two approaches

6 Conclusions

The system presented provides a useful and accurate method for translating the annotation of the Penn Treebank into a CG annotation. Comparisons with an alternative approach suggest that the increase of control provided by the system lead to a more accurate and compact translation, which is more linguistically plausible. Most importantly, the system is flexible enough to allow the user to annotate corpora with the kind of CG they are interested in, which is vital when it is to be used for evaluation.

It would be useful to expand the systems to work on the full treebank i.e. including sentences with movement (see Hockenmaier *et al* (Hockenmaier et al., 2000) for discussion of a possible method). The correcting of the annotation of the treebank during translation should also be investigated further.

References

Y. Bar-Hillel, C. Gaifman, and E. Shamir. 1964. On categorial and phrase structure grammars. In *Language and Information*, pages 99 – 115. Addison-Wesley. First appeared in The Bulletin of the Research Council of Israel, vol. 9F, pp. 1-16, 1960.

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre, 1994. *Bracketing Guidlines for Treebank II Style Penn Treebank Project*.

Rens Bod. 1995. *Enriching Linguistics with Statistics: Performance Models of Natural Language*. Ph.D. thesis, Department of Computational Linguistics, Universiteit van Amsterdam.

Eugene Charniak. 1996. Treebank grammars. In *Proceedings of AAAI/IAAI*.

John Chen and K. Vijay-Shanker. 2000. Automated extraction of tags from the Penn Treebank. In *Proceedings of the 6th International Workshop on Parsing Technologies*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Julia Hockenmaier, Gann Bierner, and Jason Baldridge. 2000. Providing robustness for a ccg system. In *Proceedings of the Workshop on Linguistic Theory and Grammar Implementation, ESSLLI 2000*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Alexander Krotov, Mark Hepple, Robert Gaizauskas, and Yorick Wilks. 1998. Compacting the Penn Treebank grammar. In *Proceedings of COLING'98–ACL'98*.

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

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *The ARPA Human Language Technology Workshop*.

Günter Neumann. 1998. Automatic extraction of stochastic lexicalized tree grammars from treebanks. In *Proceedings of the 4th Workshop on tree-adjoining grammars and related frameworks*.

Mark Steedman. 1993. Categorial grammar. *Lingua*, 90:221 – 258.

Stephen Watkinson and Suresh Manandhar. 2000. Unsupervised lexical learning with categorial grammars using the LLL corpus. In James Cussens and Sašo Džeroski, editors, *Learning Language in Logic*, volume 1925 of *Lecture Notes in Artificial Intelligence*. Springer.

Stephen Watkinson and Suresh Manandhar. 2001. A psychologically plausible and computationally effective approach to learning syntax. To appear at CoNLL'01.

Mary McGee Wood. 1993. *Categorial Grammars*. Linguistic Theory Guides. Routledge. General Editor Richard Hudson.

F. Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*.