Extending the Coverage of a CCG System

JULIA HOCKENMAIER

ICCS, Division of Informatics, University of Edinburgh, Edinburgh EH8 9LW, UK. E-mail: julia@cogsci.ed.ac.uk

GANN BIERNER

ICCS, Division of Informatics, University of Edinburgh, Edinburgh EH8 9LW, UK. E-mail: gbierner@cogsci.ed.ac.uk

JASON BALDRIDGE

ICCS, Division of Informatics, University of Edinburgh, Edinburgh EH8 9LW, UK. E-mail: jmb@cogsci.ed.ac.uk

ABSTRACT: We demonstrate ways to enhance the coverage of a symbolic NLP system through data-intensive and machine learning techniques, while preserving the advantages of using a principled symbolic grammar formalism. We automatically acquire a large syntactic CCG lexicon from the Penn Treebank and combine it with semantic and morphological information from another hand-built lexicon using decision tree and maximum entropy classifiers. We also integrate statistical preprocessing methods in our system.

KEYWORDS: *CCG, Categorial Grammar, Treebank, Lexicon Extraction, Semantics, Maximum Entropy, Decision Trees*

1 Introduction

In this paper, we present a way of extending the lexical coverage of an existing Combinatory Categorial Grammar (CCG) system. We automatically acquire a large syntactic CCG lexicon from the University of Pennsylvania (Penn) Treebank [21], and demonstrate how this syntactic lexicon can be combined with further morphological information and simple semantic interpretations from a hand-built lexicon. We also show how statistical preprocessing methods such as sentence detection and tokenization are integrated into our system.

Like most research in data-intensive NLP, our goal is to build a system which does not suffer from the fragility (and expense) of many hand-built systems. At the same time we want our system to benefit from the use of a principled grammar formalism like CCG. Our hybrid approach is motivated by these goals. Since the syntactic lexicon has been acquired from the Treebank, it is guaranteed to have a wide coverage of constructions occurring in real newspaper text. As we use categorial grammar, supplementing this syntactic lexicon with semantic information allows us to generate logical forms and do further semantic analyses. Together with the lexicon, the preprocessing components allow us to parse free text.

The techniques we discuss are useful to other symbolic formalisms such as Tree-Adjoining Grammar (TAG, [17]), Head-driven Phrase Structure Grammar (HPSG, [24]), and Lexical-Functional Grammar (LFG, [19]). Like TAG, CCG is a fully lexicalized, mildly context-sensitive formalism with worst-case polynomial time parsing [35]. Being fully lexicalized, both TAG and CCG have an essentially invariant rule component, which reduces the task of grammar development mostly to that of creating a lexicon. CCG has further benefits, such as semantic transparency and an elegant treatment of coordination. Also, CCG encodes subcategorization information and syntactic potential in a compact and simple representation. Finally, as Doran et al. [12] point out, CCG lexicons are generally more compact than TAG lexicons.

Efforts have been made to create wide-coverage lexicons for both TAG and CCG. For example, the XTAG project [40] has built a substantial English TAG lexicon which covers an extensive number of English constructions. In addition, Xia [39], Chiang [8] and Chen and Vijay-Shanker [7] have acquired TAG lexicons from the Penn Treebank [21].

Doran and Srinivas [12] induced a CCG lexicon from the XTAG English grammar by mapping TAG trees to CCG categories. Villavicencio [36] did a semi-automatic translation of the Alvey Natural Language Tools English grammar [15] to create a large CCG lexicon. Recently, Watkinson and Manandhar [37] suggested an alternative method for extracting categorial grammar lexicons from the Penn Treebank (see §5.14).

We begin with a brief introduction to our system and to CCG. Next, we give a description of a lexicon which contains hand-coded linguistic specifications for closed lexical classes and basic category and semantic frame information for open ones. Then in §5 and §6 we describe how we acquire a much larger syntactic lexicon and then endow it with the morphological and semantic information from the original lexicon. In §7, we discuss how we can make the system more robust through well known, statistical preprocessing techniques. We end with conclusions and an outlook on future work.

2 The Grok Library

Our current system is constructed with components from the Grok library, which began its life as a small CCG system for implementing and testing syntactic and semantic analyses. Grok is now a maturing Java library of natural language components. Its parsing subsystem provides extensive support for building CCG lexicons and using them to parse text. It also includes a number of preprocessing components as described in §7 as well as limited facilities for semantic and pragmatic analysis, generation, and hooks for working with other systems such as the Festival speech synthesis system.

The feasibility of using Grok for natural language applications has been demonstrated in [4]. It has also been used to create a natural language agent which has been connected to NASA's spoken natural language dialogue system ([27]; the standard natural language agent for this system is Gemini [13]).[1] The material presented in this paper is part of the measures we are taking towards improving Grok as a practical basis for building robust NLP systems.

We do not discuss Grok in great detail here, and instead refer the interested reader to the Grok homepage (`http://grok.sourceforge.net`) and [4] for more information. Grok and the maximum entropy package discussed later are both distributed under the GNU Lesser General Public License [14] and are available for download at the following addresses:

```
http://grok.sourceforge.net
http://maxent.sourceforge.net.
```

3 Combinatory Categorial Grammar

CCG was introduced by Ades and Steedman [1] as a generalization of categorial grammar, which was originally developed by Ajdukiewicz [2] and Bar Hillel [3]. Categorial grammars consist of two main parts: a set of rules and a lexicon of words and their associated syntactic types and semantic interpretations. Examples of lexical entries are given in (1).

(1) a. **John** $\vdash$ NP: `john`

[1] This was work carried out by one of the authors, Jason Baldridge, while he was a visiting student researcher at NASA's Research Institute for Advanced Computer Science.

b. **spam** $\vdash$ NP: $\mathtt{spam}$

c. **likes** $\vdash$ $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$: $\lambda x.\lambda y.\mathtt{like}(y,x)$

The lexical item appears to the left of the turnstile '$\vdash$', and the right side specifies the syntactic type and the semantic interpretation separated by the colon ':'. The backward '\' and forward '/' slashes in the syntactic category of (1c) indicate the directions (left and right, respectively) in which the category for the verb **likes** must find its NP arguments. If a word can have a category X/Y as well as a category $\mathsf{X}\backslash\mathsf{Y}$, the nondirectional slash variable '|' can be used to abbreviate this to one category $\mathsf{X} \mid \mathsf{Y}$.

We follow [32] in using the lambda notation when describing the semantics of lexical entries and combinatory rules.

During derivations, categories are retrieved from the lexicon and put together by the rules of CCG. The two basic rules are those of forward and backward function application, given in (2). These are the only rules allowed in the form of categorial grammar introduced by Ajdukiewicz and Bar Hillel (AB categorial grammar).

(2) a. *Forward Application* (>):
$$\mathsf{X}/\mathsf{Y} : f \quad \mathsf{Y} : a \quad \Rightarrow \quad \mathsf{X} : fa$$
b. *Backward Application* (<):
$$\mathsf{Y} : a \quad \mathsf{X}\backslash\mathsf{Y} : f \quad \Rightarrow \quad \mathsf{X} : fa$$

The juxtaposition fa indicates the application of the semantic function f to the argument a. With these rules and with the lexical entries given in (1), the following derivation may proceed, yielding the category S (a sentence) from the string *John likes spam*, with the appropriate interpretation.

(3)
$$\dfrac{\dfrac{\textbf{John}}{\mathsf{NP} : \mathtt{john}} \quad \dfrac{\dfrac{\textbf{likes}}{(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.\mathtt{like}(y,x)} \quad \dfrac{\textbf{spam}}{\mathsf{NP} : \mathtt{spam}}}{\mathsf{S}\backslash\mathsf{NP} : \lambda y.\mathtt{like}(y,\mathtt{spam})}>}{\mathsf{S} : \mathtt{like}(\mathtt{john},\mathtt{spam})}<$$

Apart from the basic rules of function application, CCG also allows three classes of so-called combinatory rules, each of which corresponds to one of the simplest combinators of [11]. The three combinators incorporated into CCG are composition, type-raising, and substitution, abbreviated as **B**, **T**, and **S** respectively. There is also a syncategorematic rule for coordination, <Φ>.

(4) Rules corresponding to the composition combinator **B**.

a. *Forward Composition* (>**B**):
$$\mathsf{X}/\mathsf{Y} : f \quad \mathsf{Y}/\mathsf{Z} : g \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}/\mathsf{Z} : \lambda x.f(gx)$$
b. *Forward Crossing Composition* (>$\mathbf{B}_\times$):
$$\mathsf{X}/\mathsf{Y} : f \quad \mathsf{Y}\backslash\mathsf{Z} : g \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}\backslash\mathsf{Z} : \lambda x.f(gx)$$

c. *Backward Composition* ($<\mathbf{B}$):
$\mathsf{Y}\backslash\mathsf{Z} : g \quad \mathsf{X}\backslash\mathsf{Y} : f \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}\backslash\mathsf{Z} : \lambda x.f(gx)$

d. *Backward Crossing Composition* ($<\mathbf{B}_{\times}$):
$\mathsf{Y}/\mathsf{Z} : g \quad \mathsf{X}\backslash\mathsf{Y} : f \quad \Rightarrow_{\mathbf{B}} \quad \mathsf{X}/\mathsf{Z} : \lambda x.f(gx)$

(5) Rules corresponding to the type-raising combinator $\mathbf{T}$.[2]

a. *Forward Type-raising* ($>\mathbf{T}$):
$\mathsf{X} : a \quad \Rightarrow_{\mathbf{T}} \quad \mathrm{T}/(\mathrm{T}\backslash\mathsf{X}) : \lambda p.pa$

b. *Backward Type-raising* ($<\mathbf{T}$):
$\mathsf{X} : a \quad \Rightarrow_{\mathbf{T}} \quad \mathrm{T}\backslash(\mathrm{T}/\mathsf{X}) : \lambda p.pa$

(6) Rule corresponding to the substitution combinator $\mathbf{S}$.

a. *Forward Substitution* ($>\mathbf{S}$):
$(\mathsf{X}/\mathsf{Y})/\mathsf{Z} : f \quad \mathsf{Y}/\mathsf{Z} : g \quad \Rightarrow_{\mathbf{S}} \quad \mathsf{X}/\mathsf{Z} : \lambda x.fx(gx)$

b. *Forward Crossing Substitution* ($>\mathbf{S}_{\times}$):
$(\mathsf{X}/\mathsf{Y})\backslash\mathsf{Z} : f \quad \mathsf{Y}\backslash\mathsf{Z} : g \quad \Rightarrow_{\mathbf{S}} \quad \mathsf{X}\backslash\mathsf{Z} : \lambda x.fx(gx)$

c. *Backward Substitution* ($<\mathbf{S}$):
$\mathsf{Y}\backslash\mathsf{Z} : g \quad (\mathsf{X}\backslash\mathsf{Y})\backslash\mathsf{Z} : f \quad \Rightarrow_{\mathbf{S}} \quad \mathsf{X}\backslash\mathsf{Z} : \lambda x.fx(gx)$

d. *Backward Crossing Substitution* ($<\mathbf{S}_{\times}$):
$\mathsf{Y}/\mathsf{Z} : g \quad (\mathsf{X}\backslash\mathsf{Y})/\mathsf{Z} : f \quad \Rightarrow_{\mathbf{S}} \quad \mathsf{X}/\mathsf{Z} : \lambda x.fx(gx)$

(7) The syncategorematic rule for coordination $<\Phi>$.

a. *Coordination* $<\Phi>$:
$\mathsf{X} : f \quad \mathsf{conj} : b \quad \mathsf{X} : g \quad \Rightarrow_{<\Phi>} \quad \mathsf{X} : \lambda\ldots b(f\ldots)(g\ldots)$

These combinatory rules can be restricted to given types for a given language. They bring CCG's weak generative capacity from the context-free level of AB categorial grammars to mild context-sensitivity [18]. This allows a wider range of syntactic analyses, including the famous Dutch crossing dependencies [5, 30, 34]. They also allow derivations to proceed in many semantically equivalent ways, as the following alternative to derivation (3) demonstrates.

(8)
$$\begin{array}{ccc}
\textbf{John} & \textbf{likes} & \textbf{spam} \\
\hline
\mathsf{NP} : \mathtt{john} & (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.\mathtt{like}(y,x) & \mathsf{NP} : \mathtt{spam} \\
\hline_{>\mathbf{T}} & & \\
\mathrm{T}/(\mathrm{T}\backslash\mathsf{NP}) : \lambda p.p\ \mathtt{john} & & \\
\hline_{>\mathbf{B}} & & \\
\multicolumn{2}{c}{\mathsf{S}/\mathsf{NP} : \lambda x.\mathtt{like}(\mathtt{john},x)} & \\
\hline_{>} & & \\
\multicolumn{3}{c}{\mathsf{S} : \mathtt{like}(\mathtt{john},\mathtt{spam})}
\end{array}$$

Here we derive the same result as before, but with a different derivational history. It should be noted that, unlike many formalisms, syntactic derivation is not considered a representational level in CCG. Rather, it is only a record of

[2] T in these rules is a variable over categories.

the steps which were taken in combining the categories which were introduced into the derivation. This flexibility plays a major role in CCG's extensive coverage of coordination phenomena, its account of intonation, and its capacity for incremental processing. Extensive discussions of these analyses as well as general introductions to the linguistic properties and motivations of the CCG formalism can be found in [31, 32, 33, 34].

4 A Hand-Built Lexicon for CCG

We begin by describing the underspecified, hand-built lexicon which provides our system with semantic and morphological information. In §6, we show how this information is combined with the syntactic lexicon which we acquire from the Treebank.

The entries in the hand-built lexicon pair syntactic categories with semantic representations. For closed class words such as prepositions and determiners, the lexicon contains specific entries, whereas it contains generic (underspecified) entries for open class items such as nouns or verbs. We use the notion of *families* from XTAG [40] to organize the open class part of the lexicon. Each family is marked with an associated part of speech, and contains one or more generic entries. In total, there are 75 generic entries for 29 families. There are families for nouns, pronouns, verbs with a variety of subcategorization frames, adjectives, comparatives, adverbs, modals, small clauses, wh-words, prepositions, conjunctions, and more. Examples for such open class families with their associated generic entries for are given in (9)[3]. Bierner gives a more detailed description of many of these in [4].

The semantics for these generic lexical entries is given in the form of templates, which leave the predicate, P, unspecified.

(9)	a. Intransitive:	V	$\mathsf{S\backslash NP}$	$\lambda x.P(x)$
	b. Transitive:	V	$\mathsf{S\backslash NP/NP}$	$\lambda x \lambda y.P(y,x)$
	c. Ditransitive:	V	$\mathsf{S\backslash NP/NP/NP}$	$\lambda x \lambda y \lambda z.P(z,y,x)$
			$\mathsf{S\backslash NP/PP/NP}$	$\lambda x \lambda y \lambda z.P(z,x,y)$
	d. Sent Comp:	V	$\mathsf{S\backslash NP/S_{ind\mid int}}$	$\lambda p \lambda x.P(x,p)$
			$\mathsf{S\backslash S\mid NP}$	$\lambda x \lambda p.P(x,p)$
	e. Control:	V	$\mathsf{S\backslash NP/(S_{to}\backslash NP)}$	$\lambda p \lambda x.P(x,p)$
	f. Noun:	N	$\mathsf{NP_{bare+,comp-}}$	$\lambda x.P(x)$
	g. Adjective:	A	$\mathsf{NP_{comp-}/NP_{bare+}}$	$\lambda p \lambda x.P(x) \wedge p(x)$

We also use the XTAG morphological database to provide us with the part of speech, stem, and other morphological features of open class words:

[3] This list of families is an example and is not complete. We have also only included some of the more important features on the categories. In addition, some families have more entries than are shown.

(10)	**walks**:	`V`	walk	3sg pres
	walked:	`V`	walk	past
		`V`	walk	pparticiple

The part of speech is used to identify the families an open class word belongs to. Since part of speech information alone does not distinguish between different subclasses of words, such as transitive or intransitive verbs, open class words are assumed to belong to all families with the parts of speech suggested by the morphological analyzer, and are given every category in those families. For instance, the morphological entries in (10) associate **walked** with the part of speech tag `V`. **Walked** therefore belongs to all families with the tag `V` and is given every entry of those families. These are the intransitive, transitive, ditransitive, sentential complement, and control families shown in (9).

Specific entries for given words are created by instantiating all entries given in the corresponding families. The predicate variable P in the semantic template is instantiated with the stem of the item. For example, the semantics of intransitive verbs is $\lambda x.P(x)$. When creating an entry for **walks**, P is bound to `walk`: $\lambda x.\mathtt{walk}(x)$.

We also percolate the features retrieved from the morphological database into the lexicon:

(11) **walks** $\vdash \mathsf{S}\{\mathsf{tense} = \mathsf{pres}\}\backslash\mathsf{NP}\{\mathsf{num} = \mathsf{s}, \mathsf{per} = 3\}$: $\lambda x.\mathtt{walk}(x)$
walked $\vdash \mathsf{S}\{\mathsf{tense} = \mathsf{past}\}\backslash\mathsf{NP}$: $\lambda x.\mathtt{walk}(x)$
walked $\vdash \mathsf{S}\{\mathsf{tense} = \mathsf{ppart}\}\backslash\mathsf{NP}$: $\lambda x.\mathtt{walk}(x)$

Used alone, this hand-built lexicon achieves wide coverage, but at the price of high ambiguity and over-generation. For example, the word **devoured** will be given an intransitive entry even though it can only appear in a transitive context. However, this is not of great concern since this lexicon is not intended to be used on its own. In §6, we show how we further restrict this lexicon using the acquired lexicon discussed in §5 so that we may improve the accuracy and efficiency of parsers which use it.

5 The Acquired Lexicon

In this section, we describe how categorial lexicons can be acquired from the University of Pennsylvania Treebank [21]. We then analyze a lexicon which has been extracted from sections 02-21 of the Wall Street Journal subcorpus of the Treebank and compare it to a lexicon extracted from the Brown subcorpus.

The Wall Street Journal subcorpus of the Penn Treebank contains 1 million words of parsed and tagged Wall Street Journal text collected in 1989. The Brown subcorpus contains approximately 395,000 words of lore and fiction. Constituents are enclosed in brackets, with a label indicating the part of speech tag or syntactic category. A typical example is shown in (12).

(12)
```
(S (PP-TMP (IN In)
     (NP (DT the) (NN past) (NN decade)))
   (, ,)
   (NP-SBJ (JJ Japanese) (NNS manufacturers))
   (VP (VBD concentrated)
     (PP-CLR (IN on)
       (NP (NP (JJ domestic) (NN production))
           (PP (IN for)
               (NP (NN export))))))
   (. .))
```

In the following, we will omit part of speech tags and other irrelevant details of the trees when presenting examples.

The Treebank markup is designed so that a distinction between complements and adjuncts can be read off the labels. However, this is not always marked explicitly, and we use a heuristic procedure which uses the label of a node and its parent to make this distinction.

Though the Treebank does not specifically indicate syntactic heads, a deterministic procedure for identifying them is given in [10] (this was originally developed by [20]), and we use a slightly modified version of this heuristic. Additionally, there are different types of null elements encoding traces, multiple attachments and attachment ambiguities. The presence of these null elements allows us to infer the correct categories even in relative clauses, wh-questions and coordination constructions. Our treatment of these null elements is discussed in §5.2 - §5.7.

To acquire the lexicon, we have developed an algorithm which annotates the nodes in the trees with CCG categories in a top-down recursive manner, corresponding to a reverse CCG derivation. The resulting categories for the leaf nodes constitute the lexicon. Our algorithm is very similar to [39], [7] and [8], who have developed different ways of extracting TAG trees from the Treebank. But since all of these algorithms rely crucially on head-finding procedures and heuristics to distinguish complements from adjuncts, it is possible that different implementations of the same algorithm would yield very different lexicons, thus making a direct comparison of the published results very difficult (see [7] for the impact of different complement/adjunct heuristics). Recently, [37] suggested an alternative way of extracting AB categorial grammar lexicons from a subcorpus of the Penn Treebank without null elements. This approach is discussed in §5.14.

We assume the atomic categories S, NP and PP, and employ features to distinguish between declarative sentences ($\mathsf{S_{decl}}$), wh-questions ($\mathsf{S_{whq}}$), yes-no questions ($\mathsf{S_{q}}$), embedded declaratives ($\mathsf{S_{emb}}$) and embedded questions ($\mathsf{S_{qemb}}$).[4]

[4] Note that in this paper we will sometimes abbreviate the verb phrase category $\mathsf{S\backslash NP}$ as VP. However, this is only to improve readability, and a category such as $\mathsf{VP_{decl}}$ always stands for $\mathsf{S_{decl}\backslash NP}$.

We also distinguish different kinds of verb phrases ($\mathsf{S\backslash NP}$), such as bare infinitives, to-infinitives, past participles in normal past tense, present participles, and past participles in passive verb phrases. This information is encoded as an atomic feature on the category, *e.g.* $\mathsf{S_{pass}\backslash NP}$ for a passive VP, or $\mathsf{S_{decl}}$ for a declarative sentence.[5] The main purpose of these features is to specify subcategorization information—for instance, the infinitival particle **to** takes a bare infinitive as its argument and yields a to-infinitive: **to** $\vdash \mathsf{(S_{to}\backslash NP)/(S_b\backslash NP)}$. The complementizer **that** takes a declarative sentence, and yields an embedded declarative: **that** $\vdash \mathsf{S_{emb}/S_{decl}}$. Following [4], we also distinguish bare and non-bare noun phrases. Determiners, such as **the**, are functions from bare to non-bare noun phrases: **the** $\vdash \mathsf{NP_{bare-}/NP_{bare+}}$. Plural nouns are always bare: **researchers** $\vdash \mathsf{NP_{bare+}}$. Apart from determiners, no other categories in the acquired lexicon specify the bareness of their noun phrase arguments. However, the feature system of the extracted lexicon alone is not as complete as the feature system of the CCG lexicon of [12] which was translated from the XTAG system. But, as described in §4 and §6, we use the features given in the XTAG morphological database in our hand-build lexicon, which is then used to supplement the automatically acquired lexicon with this morphological information and with semantic interpretations.

5.1 The Basic Algorithm

We will first demonstrate the basic algorithm, and then show how the different kinds of null elements in the Treebank are treated. Figure 1 gives the outline of the algorithm. We process the trees in the corpus by first determining the constituent type of each node and then assigning categories to the nodes in the trees, depending on their constituent type and Treebank label. Once we have processed the entire corpus, we create the lexicon from the word-category pairs observed on the leaf nodes in the corpus. Additionally, the frequency counts of these observed word-category pairs can be used to compute lexical unigram probabilities of the form $P(\mathit{category} \mid \mathit{word})$ or $P(\mathit{word} \mid \mathit{category})$.

If there are no null elements, we use a simple recursive top-down procedure to annotate the nodes in a tree with the appropriate categories. We traverse each local tree in the following canonical, outside-inside order: first, we assign categories to the the children to the left of the head going from left to right, then we assign categories to the children to the right of the head, going from right to left:

(13)

Root

1 ... → ... *n* *head* *m* ... ← ... *n*+1

[5] We omit these features in most of the example derivations given in this paper.

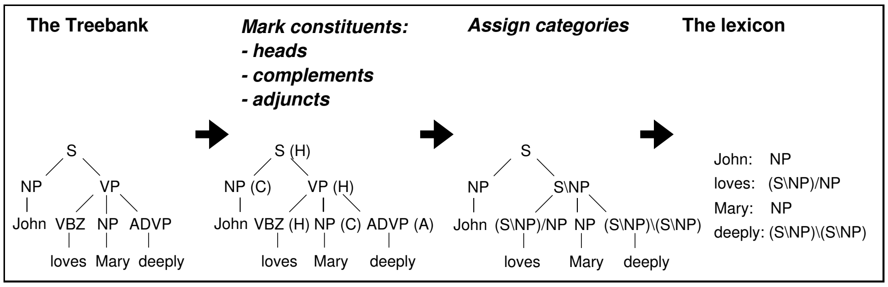


FIG. 1. Acquisition of a lexicon from the Treebank

This corresponds to a reverse derivation which respects the canonical order in which we traverse the tree. During this process, the category of the head child is built up in the following manner: the innermost result category of the head child is the category of the parent. Complements to the left of the head child are added (in the order of the tree traversal) as backward arguments, complements to the right as forward arguments. Consider the example in Figure 1. Since the head child (the VP-node) has one complement sister with category NP to its left and its parent is S, it receives the category S\NP. The VBZ-node has one complement sister, NP, to its right and a parent with category S\NP. Hence it receives category (S\NP)/NP.

Note that in the case of coordination (or lists), there can be more than one head child. We assume that in those cases, all head children have the same category (see §5.10 for a discussion of unlike coordinate phrases UCP).[6]

The category of a complement child is defined by a mapping from Treebank labels to the atomic categories. In the case of our example in Figure 1, the complements **John** and **Mary** are given the category NP.

The category of an adjunct node is a unary functor X/X or X\X, whose argument and result category are identical. The directionality of the argument is determined by the relative position of the adjunct: if it appears to the left of the head child, it is a forward looking functor, otherwise it is a backward looking functor. The category X is determined by the current head category. If we used a version of categorial grammar without composition, X would have to be equal to the current head category, but in the case of adjuncts of adjuncts, this would lead to a proliferation of categories. However, if we assume that adjuncts can combine with the heads using (generalized) composition, X can be the current head catgory with the outermost arguments stripped off. We therefore use the following strategy: A left adjunct, X/X, can combine with the head using (generalized) forward non-crossing composition (4a). Forward crossing

[6] We have developed various heuristics to identify coordination cases, but do not have space to explain them in detail.

composition (4b) is not permitted in English, since it would lead to greater freedom in word order than English allows. Hence, in the case of forward-looking adjuncts, X is the current head category minus all outermost forward-looking arguments. X/X can then combine with the current head category through (generalized) forward non-crossing composition. If we stripped off any backward-looking arguments, X/X could only combine with the head through forward crossing composition.[7].

In the case of backward-looking adjuncts, X\X, we strip off from the current head category all outermost arguments which have the same directionality as the last argument in order to obtain X – that is, if the outermost argument of the current head category is forward-looking, then we can strip off all outermost forward arguments (corresponding to generalized backward crossing composition). If the outermost argument is backward-looking, we can strip off all outermost backward arguments (generalized backward non-crossing composition).

In the case of VP-adjuncts, however, we stipulate that we do not generalize beyond the S\NP level, since we want to distinguish verbal adjuncts from sentential adjuncts. Consider for instance the adjunct **deeply** in Figure 1. Since its parent's category is S\NP and it appears to the right of the head verb, it receives the category (S\NP)\(S\NP).

In order to avoid further proliferation of category types, adjunct categories do not carry any morphological features. This means for instance that VP adjuncts all have the category (S\NP)\(S\NP) or (S\NP)/(S\NP) - that is, we do not distinguish between adjuncts appearing in declarative, infinitival, or passive verb phrases. **Deeply** is (S\NP)\(S\NP) regardless of whether it modifies **loves**, **loved** or **loving**.

In the following sections we explain how this basic algorithm can be extended to deal with the various kinds of null elements in the Treebank which are used to capture unbounded dependencies, relations of control or raising, and multiple attachments. We assume the linguistic anlyses of [34, 32] for these constructions. It is, however, not always the case that the Treebank trees correspond directly to the desired categorial analysis. Therefore, certain systematic changes have to be made to the trees, which we describe in §5.11.

5.2 Unbounded Dependencies

The Treebank represents wh-questions, relative clauses, topicalization of complements, tough movement and parasitic gaps in terms of movement. The "moved" constituent is co-indexed with a trace (`*T*`) which is inserted at the extraction site, as in (14).

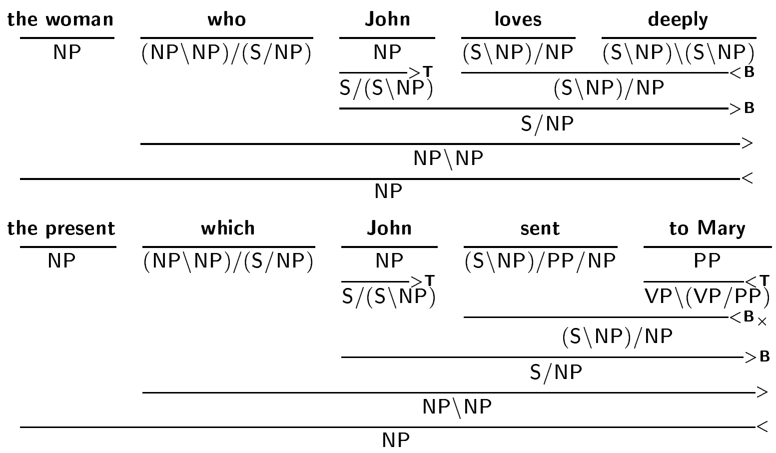
(14) `(NP-SBJ (NP The woman))`

[7] We are following a strict interpretation of the $-convention in [34].

```
(SBAR (WHNP-1 who)
      (S (NP-SBJ John)
         (VP (VBZ loves)
             (NP (-NONE-  *T*-1)))
                (ADVP deeply))))
```

CCG has a different, but similarly uniform treatment of these constructions. What in transformational terms is described as the moved constituent is analyzed in CCG as a functor over a sentence missing a complement. For instance, the relative pronoun in the following examples has the category $\mathsf{(NP\backslash NP)/(S/NP)}$, while the verbs **loves** and **sent** maintain their respective canonical categories $\mathsf{(S\backslash NP)/NP}$ and $\mathsf{((S\backslash NP)/PP)/NP}$[8]:

(15) a.



CCG allows the subject noun phrase and the incomplete verb phrase to combine via type-raising and forward composition to form a constituent with the category $\mathsf{S/NP}$, which can in turn be taken as an argument of the relative pronoun. As the relative clause itself is a noun phrase modifier, the relative pronoun has the category $\mathsf{(NP\backslash NP)/(S/NP)}$. This treatment of "movement" in terms of functors over "incomplete" constituents allows CCG to keep the same category for the verb even when its arguments are extracted.

The `*T*` traces in the Treebank help us in two ways to obtain the correct categorial analysis: firstly, their presence indicates a complement which needs to be taken into account in order to assign the correct category to the verb, and secondly, we can use a mechanism very similar to slash-feature passing in GPSG to obtain the correct category for the wh-word.

In order to obtain the correct category for the verb within the relative clause, the null element is treated as a normal constituent within its local tree. In the example above, the `*T*` trace appears under an `NP`, and therefore stands for a

[8] We use VP to abbreviate $\mathsf{S\backslash NP}$

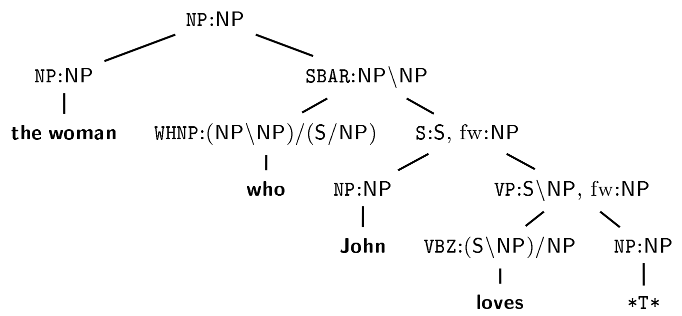
complement. This means that **loves** should receive the category (S\NP)/NP, not S\NP.

Also, we need to pass the information about the "missing" noun phrase up to the maximal projection of the sentence in which it occurs, so that the relative pronoun takes **S/NP** as an argument, and not **S**. As we do not wish to assign categories to nodes twice, the tree is traversed and searched for `*T*`-traces before the actual category assignment. If a `*T*` trace is found and appears in complement position (as determined by the label of its maximal projection), a "slash category" is passed up to the maximal projection of the sentence in which the trace occurs (here the `S`-node), hence signalling an incomplete constituent. This slash category consists of the complement category assigned to the null element (usually NP), and its direction ('forward' or 'backward', depending on the null element's position relative to the head child).

In the next step, we proceed with the category assignment as before. However, if a complement node has a slash category, the procedure is slightly different. The category of the complement node is determined by its Treebank label in the usual manner, so that the category assignment within the subtree dominated by this node is unaffected by the slash category. But when we build up the category of the head child, we do not add the (atomic) category of the complement node as an argument. Instead, we add a complex category whose argument is the slash category and whose result is the category of the complement.

In the relative clause example, the parent node (`SBAR`) is an adjunct to the `NP`, and therefore has category **NP\NP**. Within the `SBAR`, the relative pronoun (`WHNP`) is the head child, and the `S`-node is a complement, carrying a slash category "forward-NP". The category of the `S`-node itself is **S**, but, as it carries the slash category, the head category is (**NP\NP**)/(**S/NP**), not (**NP\NP**)/**S**. This results in the decorated phrase structure tree given in (16). Note that the lexical categories are the same as given in derivation (15a).

(16)



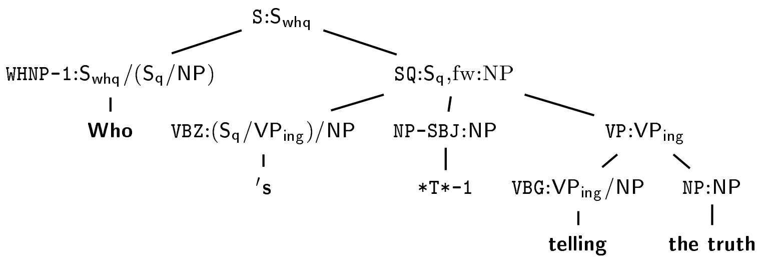
The same approach works for wh-questions. Consider the following example:

(17)
```
(TOP (SBARQ (WHNP-1 (WP Who))
            (SQ (VBZ 's)
```

```
        (NP-SBJ (-NONE- *T*-1))
        (VP (VBG telling)
            (NP the truth)))
    (. ?)))
```

In this case, the slash category "forward NP" is percolated up to the SQ-node, resulting in the following annotated tree:[9]

(18)

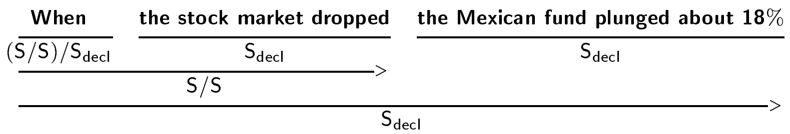


T-traces can also stand for an adjunct (here indicated by the label ADVP-TMP):

(19)
```
(TOP (S (SBAR-TMP (WHADVP-1 (WRB When))
                  (S (NP-SBJ the stock market)
                     (VP (VBD dropped)
                         (ADVP-TMP (-NONE- *T*-1)))))
        (S (NP-SBJ the Mexico fund)
           (VP plunged about 18%))
        (. .)
```

In that case, we do not need to percolate a slash category up, as the phrase **the stock market dropped** forms a complete sentence:

(20)



Tough movement is also annotated using *T*-traces:

(21)
```
(S (NP-SBJ (PRP It))
   (VP (VBZ is)
       (ADJP-PRD (JJ difficult)
                 (SBAR (WHNP-1 (-NONE- 0))
                       (S (NP-SBJ (-NONE- *))
                          (VP (TO to)
                              (VP (VB justify)
                                  (NP (-NONE- *T*-1))))))))
```

Following [32], this sentence has the following categorial analysis:[10]

---

[9] We use $\mathsf{VP_{ing}}$ to abbreviate $\mathsf{S_{ing}\backslash NP}$.

[10] We use $\mathsf{VP_{to}}$ and $\mathsf{VP_b}$ to abbreviate $\mathsf{S_{to}\backslash NP}$ and $\mathsf{S_b\backslash NP}$.

(22)

$$
\begin{array}{lllll}
\textbf{It} & \textbf{is} & \textbf{difficult} & \textbf{to} & \textbf{justify} \\
\hline
\mathsf{NP} & (\mathsf{S_{decl}}\backslash\mathsf{NP})/(\mathsf{S_{adj}}\backslash\mathsf{NP}) & (\mathsf{S_{adj}}\backslash\mathsf{NP})/(\mathsf{VP_{to}}/\mathsf{NP}) & \mathsf{VP_{to}}/\mathsf{VP_b} & \mathsf{VP_b}/\mathsf{NP} \\
 & & & \multicolumn{2}{c}{\overline{\qquad\mathsf{VP_{to}}/\mathsf{NP}\qquad}^{>\mathbf{B}}} \\
 & & \multicolumn{3}{c}{\overline{\qquad\mathsf{S_{adj}}\backslash\mathsf{NP}\qquad}^{>}} \\
 & \multicolumn{4}{c}{\overline{\qquad\mathsf{S_{decl}}\backslash\mathsf{NP}\qquad}^{>}} \\
\multicolumn{5}{c}{\overline{\qquad\mathsf{S_{decl}}\qquad}^{<}}
\end{array}
$$

We obtain this analysis from the Treebank by percolating the forward NP slash category to the SBAR-level. See §5.6 for the treatment of infinitival verb phrases.

5.3 Topicalization

T-traces are also used for topicalization. If a constituent is topicalized, or fronted, it receives the tag -TPC, and is placed at the top level of the sentence. A coindexed *T*-trace is inserted at the canonical position of that constituent:

(23)
```
(S (NP-TPC-1 (NNS Apples))
   (, ,)
   (NP-SBJ (NNP John))
   (VP (VBZ likes)
       (NP *T*-1)))
```

Following [34], we assume that topicalised noun phrases have the category S/(S/NP):

(24)

$$
\begin{array}{lll}
\textbf{Apples,} & \textbf{John} & \textbf{likes.} \\
\hline
\mathsf{S}/(\mathsf{S}/\mathsf{NP}) & \mathsf{NP} & (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} \\
 & \overline{(\mathsf{S}/(\mathsf{S}\backslash\mathsf{NP})}^{>\mathbf{T}} & \\
 & \multicolumn{2}{c}{\overline{\qquad(\mathsf{S}/\mathsf{NP})\qquad}^{>\mathbf{B}}} \\
\multicolumn{3}{c}{\overline{\qquad\mathsf{S}\qquad}^{>}}
\end{array}
$$

We stipulate that this category, which can only be derived by non-order-preserving type-raising, can be assigned by the parser to any noun phrase in sentence initial position. Therefore, this category need not appear in the lexicon. Topicalised noun phrases (NP-TPC) receive the category NP, but in order to assign the correct category to the verb, an NP with tag -TPC is not considered a complement.

If there is a resumptive pronoun (that is, the fronting is an instance of left-dislocation), there is no coreference between the fronted element and the pronoun:

(25)
```
(S (NP-TPC (NNP John))
   (, ,)
   (NP-SBJ (PRP I))
   (VP  (VBP like)
        (NP (PRP him))
        (NP-ADV (DT a) (NN lot))))
```

In these cases, we obtain the correct lexical entries in the same manner, since it suffices to ignore the (NP-TPC) as complement.

We assume, however, that a verbs of locution which take a sentential argument can have two categories, one for the untopicalised case, and one for the sentence-topicalised case:

(26) **John says** : "**I like apples**."

(27) "**I like apples**.", **John says**.

The reason for this is that if we assumed that fronted sentences had the category S/(S/S), the main verb **like** in the fronted sentence had then the category ((S/(S/S))\NP)/NP, and it seems better to us to assume that the small group of verbs which allow this construction have two categories, (S\NP)/S, and (S\S)\NP. Therefore, S-TPC are considered complements to the main verb, whereas traces which are coindexed to an S-TPC are ignored.

5.4 Right Node Raising

Right node raising constructions such as (28) can be analyzed in CCG using the same lexical categories as if the shared complement was present in both conjuncts ([32]):

(28)
```
She   applied     for     and    won         bonus pay
---   ---------   -----   ----   ---------   ---------
NP    (S\NP)/PP   PP/NP   conj   (S\NP)/NP   NP
      ------------------->B
         (S\NP)/NP
      ------------------------------------<Φ>
                (S\NP)/NP
      ----------------------------------------------->
                        S\NP
-----------------------------------------------------<
                        S
```

In order to assign the correct lexical categories to such sentences, we need to know where the canonical location of the shared complement is, ie. that the shared constituent is interpreted as a complement of both verbs, and that sentence 28 means the same as:

(29) **She applied for bonus pay and won bonus pay**.

The Treebank adopts a standard analysis of this construction, in which the shared constituent is co-indexed with two *RNR*-traces that occur in the canonical position of the shared element:

(30)
```
((S (NP-SBJ She)
    (VP (VP (VBD applied)
          (PP-CLR (IN for)
             (NP (-NONE- *RNR*-1))))
        (CC and)
        (VP (VBD won)
```

```
      (NP (-NONE- *RNR*-1)))
    (NP-1  bonus pay)
    (PP-LOC (IN under) (NP the reform law)))
  (. .)))
```

In order to assign correct lexical categories to sentences with right-node-raising, we need to alter the algorithm slightly. The category assignment proceeds in three steps for sentences which contain *RNR*-traces:

1. When determining the constituent type of nodes: Identify all nodes which are co-indexed with *RNR*-traces (eg. NP-1). These constituents are neither heads, complements nor adjuncts, and hence will get ignored in the category assignment. *RNR*-traces themselves (or their maximal projections, here NPs) are treated like ordinary constituents, and thus they can be either heads, complements or adjuncts.
2. Assign categories to the nodes as before. Nodes which are co-indexed with *RNR*-traces (NP-1) will be ignored because they are neither heads, complements nor adjuncts. *RNR*-traces themselves will receive the category of an ordinary constituent in this canonical position.
3. If the *RNR*-traces with the same index do not have the same category, this sentence cannot be processed, as the CCG analysis predicts that both consituents in canonical position have the same category.[11] Otherwise, copy the category of the *RNR*-traces, and assign it to the co-indexed node. Then assign categories in the usual top-down manner to the subtree beneath the co-indexed node.

Ignoring the coindexed constituent **bonus pay** in the first pass guarantees that **applied** is assigned $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{PP}$, not $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}/\mathsf{PP}$. Considering the *RNR*-traces as ordinary constituents guarantees that **for** is assigned PP/NP, not PP, and **won** $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$, not $\mathsf{S}\backslash\mathsf{NP}$.

In the above example, the shared constituent is a complement. The same algorithm works if the shared constituent is an adjunct, although in that case it is not strictly necessary to use this two-pass procedure.[12]

In English it is also possible for two conjoined noun phrases to share the same head:

(31) **a U.S. and a Soviet naval vessel**

This is also annotated with *RNR*-traces.

(32)
```
(NP (NP (DT a)
        (NNP U.S.)
```

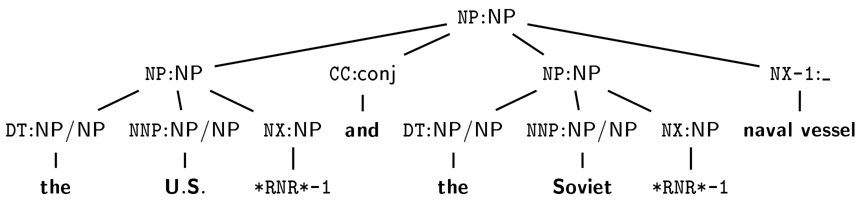
[11] This would arise if two *RNR*-trace complements did not have the same Treebank label, say if one was a PP and the other a NP, but in practice this does nott happen.

[12] Note that in the adjunct case it is conceivable that the categories of the two *RNR*-traces differ because they might be attached at different levels in the tree, but again, this does not seem to happen in practice

```
        (NX (-NONE- *RNR*-1)))
    (CC and)
    (NP (DT a)
        (JJ Soviet)
        (NX (-NONE- *RNR*-1)))
    (NX-1 (JJ naval)
          (NN vessel)))
```

Our algorithm works just as well with this case: First, the `NX-1` is ignored. Then categories are assigned to all other nodes, resulting in the following tree (without the bare noun levels inserted, see §5.11):

(33)



Then we assign NP to the shared constituent `NX-1`, and assign the corresponding categories to its daughters.

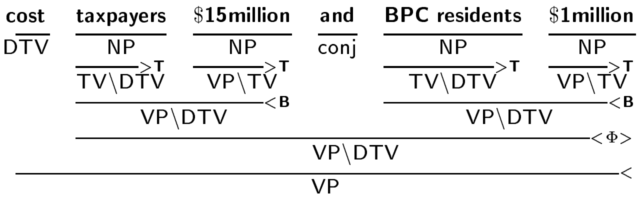
5.5 *Gapping and Argument Clusters*

If two VPs with the same head are conjoined, the second verb can be omitted:

(34) *It could cost taxpayers \$15 million and BPC residents \$1 million.*

In CCG, this is analyzed by first typeraising and composing the constituents of each argument cluster, such that the resulting category is a functor which takes a verb of the right category to its left to yield a verb phrase (cf. [34], chapter 7). Then the argument clusters are conjoined, and combine with the verb via function application:[13]

(35)



The Treebank encodes these constructions like a `VP`-coordination in which the second `VP` lacks a verb. Also, the daughters of the second conjunct are coindexed with the corresponding elements in the first conjunct using the = notation (referred to in the Treebank manual as template gapping):

[13] We use the following abbreviations: VP for S\NP, TV for transitive (S\NP)/NP and DTV for ditransitive ((S\NP)/NP/NP)

(36)
```
(S (NP-SBJ It)
   (VP (MD could)
       (VP (VP (VB cost)
               (NP-1 taxpayers)
               (NP-2 $ 15 million))
           (CC and)
           (VP (NP=1 BPC residents)
               (NP=2 $ 1 million))))
        (. .))
```

If the second VP contains constituents which do not correspond to constituents in the first VP, a null element (marked *NOT*) with the same label is inserted in the appropriate place in the first VP. This null element is coindexed with the corresponding constituent in the second VP:

(37)
```
(S-ADV (NP-SBJ (-NONE- *-1))
       (VP (VP (VBG increasing)
               (PP-DIR-2 to 2.5 %)
               (PP-TMP-3 in February 1991)
               (ADVP-TMP-4 (-NONE- *NOT*)))
           (, ,)
           (CC and)
           (VP (PP-DIR=2 to 3 %)
               (PP-TMP=3 at six month intervals)
               (ADVP-TMP=4 thereafter))))
```

We are able to assign correct categories to the constituents in the first conjunct. Since our aim is to acquire a lexicon, not to give a full categorial derivation of theses sentences, it is sufficient to assign to a constituent with a "="-index the same category as its antecedent.

A similar annotation is used for sentential gapping:

(38)
```
(S (S (NP-SBJ-1 Only the assistant manager)
      (VP (MD can)
          (VP (VB talk)
              (PP-CLR-2 (IN to)
                  (NP the manager)))))
   (CC and)
   (S (NP-SBJ=1 the manager)
      (PP-CLR=2 (TO to)
          (NP the general manager))))
```

CCG analyzes gapping with decomposition (see [34]). Again, we obtain the correct lexical categories for the constituents in the second conjunct by assigning them the same categories as the constiuents they are coindexed with.

5.6 Infinitival and Participial VPs, Gerunds

It is a design decision of the Treebank that participial phrases, gerunds, imperatives and infinitival verb phrases are annotated as sentences with a * null subject (which can be coindexed with another NP in the sentence, depending on the construction):

(39) a.
```
(NP (NP the policy)
    (S (NP-SBJ (-NONE- *))
       (VP (TO to)
           (VP (VB seduce)
               (NP socialist nations)
               (PP-CLR into the capitalist sphere)))))
```
b.
```
(S (NP-SBJ-1 The banks)
   (VP (VBD stopped)
       (S (NP-SBJ (-NONE- *-1))
          (VP (VBG promoting)
              (NP the packages))))))
```

Since VPs are analysed as $\mathsf{S\backslash NP}$s in the categorial framework, this design decision does not matter to us, since VPs in complement position receive the same category as sentences with a null subject.

5.7 Passive, Control and Raising

The Treebank also uses the null element $*$ for passive, control and raising verbs.

The surface subject of a passive sentence is co-indexed with a $*$ null element which appears in the direct object position after the past participle, for example:

(40)
```
(S (NP-SBJ-1 John)
   (VP (VBD was)
       (VP (VBN hit)
           (NP (-NONE- *-1))
           (PP (IN by)
               (NP-LGS a ball)))))
```

In this case, the null element does not indicate an argument which should be reflected in the category of the participial. Instead, the correct lexical categories are as follows:

(41) a. **was** $\vdash (\mathsf{S_{decl}\backslash NP})/(\mathsf{S_{pass}\backslash NP})$

b. **hit** $\vdash \mathsf{S_{pass}\backslash NP}$

c. **by** $\vdash ((\mathsf{S_{pass}\backslash NP})\backslash(\mathsf{S_{pass}\backslash NP}))/\mathsf{NP}$

We use the presence of the $*$ null element to distinguish the use of past participles in passive verb phrases (40) from past participles in active verb phrases such as the following example:

(42)
```
(S (NP-SBJ-1 John)
   (VP (VBZ has)
       (VP (VBN hit)
           (NP the ball))))
```

In this case, **hit** has the following lexical entry:

(43) **hit** $\vdash (\mathsf{S}_{\mathsf{pt}}\backslash\mathsf{NP})/\mathsf{NP}$

We analyse the **by**-PP in passive verb phrases as an adjunct to a passive verb phrase rather than as an argument of the passive participle. The reason for this is that the **by**-PP is optional, so we will not acquire the category $(\mathsf{S}_{\mathsf{pass}}\backslash\mathsf{NP})/\mathsf{PP}_{\mathsf{by}}$ for all passive verbs from the Treebank.

In the case of verbs like **pay for**, which subcategorize for a PP, the null element appears within the PP:

(44)
```
(S (NP-SBJ-30 (PRP$ Its)
     (NN budget))
   (VP (VBZ is)
       (VP (VBN paid)
           (PP-CLR (IN for)
                   (NP (-NONE- *-30)))
           (PP (IN by)
                   (NP-LGS (PRP you)))))
           (. .)))
```

In this example, the correct lexical categories are as follows:

(45) a. **is** $\vdash (\mathsf{S}_{\mathsf{decl}}\backslash\mathsf{NP})/(\mathsf{S}_{\mathsf{pass}}\backslash\mathsf{NP})$

b. **paid** $\vdash (\mathsf{S}_{\mathsf{pass}}\backslash\mathsf{NP})/(\mathsf{PP}/\mathsf{NP})$

c. **for** $\vdash \mathsf{PP}/\mathsf{NP}$

Note that the preposition has its ordinary category PP/NP, and that the past participle subcategorizes for the preposition alone, instead of the saturated PP. This means that in passive verb phrases with passive traces in PPs in object position, the passive trace must be taken into account as an argument to the preposition, but it must also be percolated up to the PP level in order to assign the correct category to the past participle.

Raising and subject control both have a coindexed *-trace in the subject position of the embedded clause, for instance:

(46) a.
```
(S (NP-SBJ-1 Mr. Stronach)
   (VP (VBZ wants)
       (S (NP-SBJ (-NONE- *-1))
          (VP (TO to)
              (VP (VB resume)
                  (NP a more influential role))))))
```

b.
```
(S (NP-SBJ-1 Every Japanese under 40)
   (VP (VBZ seems)
       (S (NP-SBJ (-NONE- *-1))
          (VP (TO to)
              (VP (VB be)
                  (ADJP-PRD fluent in Beatles lyrics))))))
```

Since an S with an empty subject NP has category $\mathsf{S\backslash NP}$, we obtain the correct syntactic category $(\mathsf{S_{decl}\backslash NP})/(\mathsf{S_{to}\backslash NP})$ for both **seems** and **wants**.

In the case of object control (47a), the controlled object appears as a separate argument to the verb and is coindexed with a *-trace in subject position of the complement, whereas the Treebank gives object raising (47b) a small clause analysis in which the verb takes a sentential complement:

(47) a.
```
(S (NP-TMP Last week)
   (, ,)
   (NP-SBJ housing lobbies)
   (VP (VBD persuaded)
       (NP-1 Congress)
       (S (NP-SBJ (-NONE- *-1))
          (VP (TO to)
              (VP raise the ceiling to $124,875))))
```
b.
```
(S (NP-SBJ Czechoslovakia)
   (ADVP-TMP still)
   (VP (VBZ wants)
       (S (NP-SBJ-1 the dam)
          (VP (TO to)
              (VP (VB be)
                  (VP (VBN built)
                      (NP (-NONE- *-1))))
```

As explained in §5.11, we disagree with the small clause analysis given by the Treebank, and modify the tree so that we obtain the same lexical category $(((\mathsf{S_{decl}\backslash NP})/(\mathsf{S_{to}\backslash NP}))/\mathsf{NP})$ for both verbs.

5.8 Ellipsis

The Treebank uses the null element *?* as placeholder “for a missing predicate or a piece thereof” [21]. *?* is used for VP ellipsis, and can also occur in conjunction with a VP pro-form **do**, or in comparatives:

(48) a.
```
(S (NP-SBJ No one)
        (VP (MD can)
            (VP (VB say)
                (SBAR (-NONE- *?*))))
        (. .))
```
b.
```
(S (NP-SBJ-1 Both banks)
   (VP (VBP have)
       (VP (VBN been)
           (VP (VBN battered)
               (NP (-NONE- *-1))
               (, ,)
               (SBAR-ADV (IN as)
                         (SINV (VP (VBP have)
```

```
                                  (VP (-NONE- *?*)))
                                  (NP-SBJ other Arizona banks)))
                   (, ,)
                   (PP-MNR by falling real estate prices)))))
```

c.
```
(S (NP-SBJ The total of 18 deaths)
   (VP (VBD was)
       (ADJP-PRD (ADJP far higher))
            (SBAR (IN than)
                  (S (NP-SBJ (-NONE- *))
                     (VP (VBN expected)
                         (S (-NONE- *?*))))))))
```

d.
```
(S (S (NP-SBJ You)
      (RB either)
      (VP (VBP believe)
          (SBAR Seymour can do it again)))
   (CC or)
   (S (NP-SBJ you)
      (VP (VBP do)
          (RB n't)
          (VP (-NONE- *?*)))))
```

Although the `*?*` null element indicates a semantic argument of the head of the `VP` under which it appears (*e.g.* of **expected** or **do** in the examples above), we do not reflect this argument in the syntactic category of the heads. We follow the analysis of VP ellipsis under conjunction given in [34], which argues that both conjuncts in examples such as (48d) are complete sentences. Therefore, the syntactic category of **do** is S\NP, not (S\NP)/VP. For other constructions in which `*?*` is used, a different treatment is conceivable. For instance, **as** or **than** could subcategorize for incomplete verb phrases, in which case we would have to consider `*?*` null elements complements. However, the Treebank analysis does not allow us to infer the correct features for `*?*` null elements. Therefore, we ignore any `*?*` null element.

5.9 Other Kinds of Null Elements in the Treebank

Besides the null elements discussed above, the Treebank contains further kinds of null elements, all of which we ignore when extracting a lexicon from the Treebank.

`*ICH*` ("Insert constituent here") is used for extraposition of modifiers. When there is intervening material between a modifier and the constituent it modifies, and if the intervening material causes a difference in attachment height, a `*ICH*` null element is inserted as adjunct to the modified constituent:

(49) a.
```
(S (NP-SBJ (NP Areas of the factory)
           (SBAR (-NONE- *ICH*-2)))
   (VP (VBD were)
       (ADJP-PRD particularly dusty)
```

```
            (SBAR-2 where the crocidolite was used))
            (. .))
```
b.
```
(S (NP-SBJ-130 (NP A final modification)
                (PP (-NONE- *ICH*-1)))
    (VP (VBD was)
        (VP (VBN made)
            (NP (-NONE- *-130))
            (PP-1 (TO to)
                  (NP (NP the five-point opening limit))
                      (PP for the contract)))))
        (. .))
```
c.
```
(S (NP-SBJ There)
    (VP (VBZ 's)
        (NP-PRD (NP (DT an)
                    (NN understanding)
                    (SBAR (-NONE- *ICH*-1)))
                (PP on the part of the US)
                (SBAR-1 that Japan has to expand its ''functions'' in Asia))))
```

Again, this is a case of a semantic dependency which should not be reflected in the syntactic category. Note that a constituent which is coindexed with an *ICH* null element is not a complement. We therefore treat all constituents which are coindexed with an *ICH* null element as adjuncts. In example (49a), for instance, **were** has category $(\mathsf{S}_{\mathsf{decl}}\backslash\mathsf{NP})/(\mathsf{S}_{\mathsf{adj}}\backslash\mathsf{NP})$, not $((\mathsf{S}_{\mathsf{decl}}\backslash\mathsf{NP})/\mathsf{S}_{\mathsf{emb}})/(\mathsf{S}_{\mathsf{adj}}\backslash\mathsf{NP})$.

The null element *PPA* ("Permanent Predictable Ambiguity") is used for genuine attachment ambiguities:

(50)
```
(S (NP-SBJ He)
   (ADVP-TMP already)
   (VP (VBZ has)
       (VP (VBN finagled)
           (NP (NP a $ 2 billion loan)
               (PP (-NONE- *PPA*-1)))
           (PP-CLR-1 from the Japanese government))
        (. .))
```

Since the Treebank manual states that the actual constituent should be attached at the more likely attachement site, we chose to ignore any *PPA* null element.

EXP ("Expletive") is a null element which is used in sentences with an expletive **it**. It indicates the logical subject of the clause:

(51)
```
(S (NP-SBJ (NP It)
           (S (-NONE- *EXP*-1)))
   (VP (VBZ 's)
       (ADJP-PRD too early)
       (S-1 (NP-SBJ (-NONE- *))
            (VP to say whether that will happen))))
```

Since `*EXP*` null elements do not indicate a complement to the expletive **it**, they are ignored by the program.

Another null element, `*U*`, is used to "mark the interpreted position of a unit symbol" ([21]):

(52)
```
(NP (QP ($ $) (CD 1.5) (CD billion))
    (-NONE- *U*)))
```

We ignore it, and assume the unit symbol is head. Since we lack a proper treatment of numerals, we simply assume they are all adjuncts to the unit symbol.

5.10 Fragments in the Treebank

Any constituent for which no proper analysis can be given is labelled `FRAG` (for fragment). This can be an entire tree, or part of another consituent:

(53) a. `(FRAG (NP The next province) (. ?))`

b. `(SBARQ (WRB how) (RP about) (FRAG (NP the New Guinea Fund)) (. ?)`

c.
```
(TOP (FRAG (FRAG (IN If)
                 (RB not)
                 (NP-LOC (NNP Chicago))
                 (, ,)
                 (RB then)
                 (PP-LOC (IN in) (NP New York)))
           (: ;)
           (FRAG (IN if)
                 (RB not)
                 (NP-LOC (the U.S.))
                 (, ,)
                 (RB then)
                 (NP-LOC overseas)))
           (. .)))
```

Besides `FRAG`, there are also trees which are rooted in `NP`. They are often list-like structures, or names.

(54) a.
```
(TOP (NP (NP (NNP Maxwell) (NNP R.D.) (NNP Vos))
         (NP-LOC (NP (NNP Brooklyn)) (, ,) (NP (NNP N.Y) (. .)))))
```

b.
```
(TOP (NP (NP (NP LONDON INTERBANK  OFFERED RATES)
         (: :)
         (NP (NP (NP (QP 8 3/4 %))
                 (NP (CD one) (NN month)))
             (: ;)
             (NP (NP (QP 8 3/4 %))
                 (NP (CD three) (NNS months)))
             (: ;)
             (NP (NP (QP 8 1/2 %))
                 (NP (CD six) (NNS months)))
             (: ;)
```

As can be seen from these examples, it is not always clear what a categorial analysis of such non-sentential items should be. For instance, should "**one month**" be considered a postmodifier of "**8 3/4**%"? Also, the internal structure of constituents labelled `FRAG` is often not very well analysed. Therefore we only process trees which are rooted in sentential categories (`S, SBAR, SINV, SBARQ`). Furthermore, we do not process any tree which contains a constituent labelled `FRAG`. We also have not yet implemented a proper treatment of "Unlike Coordinate Phrase" `UCP`, such as the following:

(55)
```
(PP (IN except)
    (UCP (SBAR-TMP where a court orders it)
         (CC or)
         (S-PRP to prevent the client from committing a criminal act)))
```

Discarding any tree containing constituents labelled `FRAG` or `UCP`, we still process more than 88% of the Treebank.

5.11 Changes to Treebank trees

It is not always the case that the Treebank trees correspond directly to the desired categorial analaysis. In this section, we give examples of systematic changes our program makes to Treebank trees.

The most obvious example is noun phrases. The Treebank assumes a flat internal structure with no separate noun level:

(56)
```
(NP (DT the) (NNP Dutch) (VBG publishing) (NN group))
```

As described above, we follow [4] in distinguishing bare and non-bare noun phrases. Determiners, such as **the**, are functions from bare to non-bare noun phrases: **the** $\vdash \mathsf{NP}_{\mathsf{bare}-}/\mathsf{NP}_{\mathsf{bare}+}$. Other prenominal modifiers are functions from bare to bare noun phrases, eg. **Dutch** $\vdash \mathsf{NP}_{\mathsf{bare}-}/\mathsf{NP}_{\mathsf{bare}-}$. Plural nouns are always bare: **researchers** $\vdash \mathsf{NP}_{\mathsf{bare}+}$. Apart from determiners, no other categories specify the bareness of their noun phrase arguments.

Another example is the small clause. The Treebank adopts a small clause analysis for constructions such as the following:

(57) a.
```
(S (NP-SBJ The country)
   (VP (VBZ wants)
       (S (NP-SBJ-2  half the debt)
          (VP (VBN forgiven)
              (NP (-NONE- *-2))))))
```

b.
```
(S (NP-SBJ that volume)
   (VP (VBZ makes)
       (S  (NP-SBJ it)
           (NP-PRD (NP the largest supplier))
                (PP of original TV programming)
                (PP-LOC in Europe))))
```

If these verbs occur in the passive, they are analysed as taking a small clause the subject of which is a passive NP null element (see §5.7):

(58)
```
(S (NP-SBJ-1 The refund pool)
   (VP (MD may) (RB not)
       (VP (VB be)
          (VP (VBN held)
              (S (NP-SBJ (-NONE- *-1))
                 (NP-PRD (NN hostage)))))))
```

The possibility of extractions like *"what does the country want forgiven"* suggests that these cases should rather be treated as involving two complements. We therefore eliminate the small clause, and transform the trees such that the verb takes the children of the small clause as complements. This corresponds to the following analyses:

(59) a.
```
(S (NP-SBJ the country)
   (VP (VBZ wants)
       (NP-SBJ-2  half the debt)
       (VP (VBN forgiven)
           (NP (-NONE- *-2)))))
```

b.
```
(S (NP-SBJ that volume)
   (VP (VBZ makes)
       (NP it)
       (NP-PRD (NP the largest supplier)
               (PP of original TV programming)
               (PP-LOC in Europe))))
```

c.
```
(S (NP-SBJ-1 The refund pool)
   (VP (MD may) (RB not)
       (VP (VB be)
           (VP (VBN held)
               (NP (-NONE- *-1))
               (NP-PRD hostage)))))
```

The other case where small clauses are used in the Treebank is in some constructions which are analysed as adverbial SBAR:

(60) a.
```
(S (SBAR-ADV (IN With)
             (S (NP-SBJ the limit)
                (PP-PRD in effect)))
   (, ,)
   (NP-SBJ members)
   (VP would be able to execute trades at the limit price)
   (. .))
```

b.
```
(S (SBAR-ADV (IN Though)
             (S (NP-SBJ (-NONE- *-1))
                (ADJP-PRD (JJ modest))))
       (, ,)
       (NP-SBJ-1 the change)
       (VP (VBZ reaches)
```

```
        (PP-LOC-CLR  beyond the oil patch)
        (, ,)
        (ADVP too))
    (. .)))
```

We use the same approach for these cases, and assume that the subordinating conjunction (**with** or **though**, in these examples), takes the individual constituents in the small clause as complements. In the examples above, this gives the following lexical categories:

(61) a. **with** $\vdash$ ((S/S)/PP)/NP
b. **though** $\vdash$ (S/S)/(S_{adj}\NP)

5.12 *Coverage of the Acquired Lexicon*

We acquired a reference lexicon from sections 02-21 of the WSJ subcorpus of the Treebank, which we compare in this section to a test lexicon acquired from section 23 of the WSJ. We also compare our coverage to the figures reported by Xia for a TAG lexicon extracted from the same corpus [39]. In the next section, we investigate the domain specificity of the acquired lexicons by comparing the reference lexicon to a lexicon acquired from the Brown subcorpus.

As explained in §5.10, we only process trees rooted in sentential categories (`S`, `SBAR`, `SINV`, `SBARQ`) and discard any sentences containing constituent labelled as `FRAG` or `UCP`. We still process more than 88% of the Treebank, and the results reported here refer to the processed subcorpus of the Treebank. The reference lexicon contains 70,766 entries for 41,003 word types (or 834,673 word tokens). In total, there are 1,054 category types. Table 1 shows the per token frequency distribution of the category types in the corpus. Out of the 1,054 category types, 360 occur only once. A word has on average 1.725 categories. As can be seen from table 2, there are a few words which have very many categories. Most of these are closed class items, such as different inflection forms of the copula and prepositions.

A lexicon acquired from the same subcorpus, but without any morphological features on verbal categories, has 421 category types. In this lexicon, the average number of categories per word is 1.63.

The test lexicon has 11,314 entries for 7,681 word types (or 49,490 word tokens). On average a word has 1.473 categories. There are 371 category types, out of which 16 were not in the reference lexicon. Of these new category types, one occurs twice, and the rest occur only once.

Table 3 shows the coverage of the reference lexicon on the test data, and compares our results against the figures reported by Xia [39] for a TAG-lexicon extracted from the same corpus.[14]

[14] Note that the figures are not exactly comparable, since we do not process certain types of trees in the Treebank, whereas [39] filters out invalid templates

Token frequency f	w/features	w/o features
$f = 1$	360	119
$1 < f < 10$	343	122
$10 \leq f < 100$	206	99
$100 \leq f < 1,000$	88	38
$1,000 \leq f < 10,000$	44	27
$10,000 \leq f$	13	15

TABLE 1. Frequency distribution of category types in corpus

Number of entries n	Words
$n = 1$	28,365
$1 \leq n < 10$	12,170
$10 \leq n < 20$	366
$20 \leq n < 30$	65
$30 \leq n < 40$	20
$40 \leq n$	16

TABLE 2. Distribution of number of entries per word (with features)

	CCG lexicon	TAG lexicon
Types of categories/templates	1,054	3,014
Entries also in reference lexicon:	95.09%	94.03%
Entries not in reference lexicon:	4.91%	5.97%
Known words:	2.26%	3.50%
- Known words, known categories:	2.23%	3.42%
- Known words, unknown categories:	0.03%	0.08%
Unknown words:	2.63%	2.45%
- Unknown words, known categories:	2.63%	2.46%
- Unknown words, unknown categories:	-	0.01%

TABLE 3. Coverage of WSJ lexicons on unseen WSJ data

For 95.09% of the word tokens in section 23, the corresponding entry exists in the reference lexicon. For the remaining 4.91% of the tokens, no corresponding entry can be found in the reference lexicon. Here we can distinguish between known words, which already have an entry in the reference lexicon, and unknown words, which do not occur in the reference lexicon. And, similarly, we can distinguish between known categories and unknown categories. This gives four different types of cases in which the corresponding entry does not exist in the reference lexicon. As can be seen from Figure 3, for the CCG lexicon, the most frequent of these is the case of unknown words carrying known categories. In the TAG lexicon, the most frequent type of new entries is the case of known words with known categories. This might be due to the fact that the TAG lexicon contains more elementary tree templates (3,014) than the CCG lexicon CCG categories (1,054), which by itself can be seen as further confirmation of the observation made by [12] that CCG lexicons tend to be more compact than TAG lexicons. A very high proportion of the unknown words (76.57%) are nouns or part of compound nouns, carrying the nominal categories $\mathsf{NP}_{\mathsf{bare+}}$ or $\mathsf{NP}_{\mathsf{bare+}}/\mathsf{NP}_{\mathsf{bare+}}$. This means that we can increase the coverage of our lexicon by assigning $\mathsf{NP}_{\mathsf{bare+}}$ and $\mathsf{NP}_{\mathsf{bare+}}/\mathsf{NP}_{\mathsf{bare+}}$ to all unknown words. By doing this, we increase the coverage to 97.11%. Using preprocessing tools, as described in §7, will also alleviate the coverage problem for certain classes of lexical items such as names, dates and numbers.

5.13 Domain-Specificity of the Acquired Lexicon

We also extracted a lexicon from the Brown subcorpus included in Treebank release 3. This is a corpus of general fiction and lore, and we would expect a lexicon extracted from this corpus to be substantially different from a lexicon extracted from the Wall Street Journal. The Brown corpus lexicon contains 48,815 entries for 27,384 word types (and 400,599 word tokens). On average, a word has 1.7826 categories. Table 4 shows the coverage of the WSJ reference lexicon on the Brown corpus. For 88.73% of the words in the Brown corpus, (corresponding to 43.36% of the lexical entries in the Brown lexicon), the corresponding lexical entry can also be found in the WSJ lexicon. 4.82% of the tokens created new lexical entries of seen words and seen categories (accounting for 24.10% of the lexical entries in the Brown lexicon). Assuming that unknown words can be either $\mathsf{NP}_{\mathsf{bare+}}$ or $\mathsf{NP}_{\mathsf{bare+}}/\mathsf{NP}_{\mathsf{bare+}}$, we can find the correct lexical entry for 93.29% of the tokens in the Brown corpus in the WSJ lexicon.

A comparison of the WSJ lexicon against the Brown lexicon reveals that the Brown lexicon contains the correct entries for 81.42% of the tokens in the WSJ corpus. 6.70% of the tokens in the WSJ corpus are pairs of seen words and seen categories, with the entry missing in the Brown lexicon. 0.054% of the tokens in the WSJ are pairs of seen words and unseen categories. 9.37% of the tokens which did not occur in the Brown corpus were $\mathsf{NP}_{\mathsf{bare+}}$ or $\mathsf{NP}_{\mathsf{bare+}}/\mathsf{NP}_{\mathsf{bare+}}$. Only 2.34% of the tokens were unseen words with other (previously seen) categories, and 0.0021% of the tokens were unseen words with unseen categories. This means that if we assign $\mathsf{NP}_{\mathsf{bare+}}$ or $\mathsf{NP}_{\mathsf{bare+}}/\mathsf{NP}_{\mathsf{bare+}}$ to unknown words, we can find the correct entry for 90.79% of the tokens in the Wall Street Journal from the Brown lexicon.

	Brown
Entries also in reference lexicon:	88.73%
Entries not in reference lexicon:	
- Known words, seen categories:	4.82%
- Known words, new categories:	0.2%
- Unknown words, seen categories:	6.23%
- Unknown words, new categories:	0.01%

TABLE 4. Domain-specificity: coverage of WSJ lexicon on Brown corpus

	WSJ 02-21
Entries also in reference lexicon:	81.42%
Entries not in reference lexicon:	
- Known words, seen categories:	6.70%
- Known words, new categories:	0.05%
- Unknown words, seen categories:	11.82%
- Unknown words, new categories:	0.002%
Unknown words: N or N/N	65.63

TABLE 5. Domain-specificity: coverage of Brown lexicon on the WSJ

5.14 Comparison with an Alternative Algorithm

Watkinson and Manandhar [37] present an alternative algorithm for the extraction of AB categorial lexicons from the Penn Treebank. However, they do not present a way of dealing with the various null elements in the Treebank, which means that they can only process a small subset of the sentences in the corpus.[15] Furthermore, unlike ours, their algorithm does not correspond to a reverse derivation, and therefore it is unclear how the correctness of their translation can be guaranteed unless categories assigned in the initial step can later be modified.

In particular, without such a correction, it would be possible for their method to assign lexical categories to a sentence which cannot be combined to derive a sentential category. Their algorithm proceeds in four stages:

1. Map some POS-tags to categories.
2. Look at the surrounding subtree to map other POS-tags to categories
3. Annotate subtrees with head, complement and adjunct information using heuristics similar to [10].

[15] A search with tgrep showed that out of the 49,298 sentences in the Treebank, 34,318 contain a null element matching the regular expression /\ */

4. Assign categories to the remaining words in a bottom-up fashion.

In the first step, Watkinson and Manandhar map some part-of-speech tags deterministically to CG categories. The example they give is `DT` $\rightarrow$ $\mathsf{NP/N}$. However, this analysis is only correct for determiners appearing in noun phrases in complement position. For instance, it is not the correct analysis for determiners in temporal NPs.[16] Consider the `NP-TMP` in the following example:

(62)
```
(S (NP-SBJ South Korea)
   (VP (VBZ has)
       (VP (VBN recorded)
           (NP a trade surplus)
           (NP-TMP (DT this)
                   (NN year))))
    (. .)))
```

Here is the derivation of the embedded verb phrase:

(63)
$$\begin{array}{cccc}
\textbf{recorded} & \textbf{a trade surplus} & \textbf{this} & \textbf{year} \\
\hline
\mathsf{(S_{pt}\backslash NP)/NP} & \mathsf{NP} & \mathsf{((S\backslash NP)\backslash(S\backslash NP))/N} & \mathsf{N} \\
\hline
\multicolumn{2}{c}{\mathsf{S_{pt}\backslash NP}\ >} & \multicolumn{2}{c}{\mathsf{(S\backslash NP)\backslash(S\backslash NP)}\ >} \\
\hline
\multicolumn{4}{c}{\mathsf{S_{pt}\backslash NP}\ <}
\end{array}$$

In step 3, they use very similar heuristics to ours (both are based on [10]) to identify heads, adjuncts and complements. Thus, the `NP-TMP` would be identified as an adjunct, and either the analysis given in the first step would have to be modified, or the categories cannot combine:

(64)
$$\begin{array}{cccc}
\textbf{recorded} & \textbf{a trade surplus} & \textbf{this} & \textbf{year} \\
\hline
\mathsf{(S_{pt}\backslash NP)/NP} & \mathsf{NP} & \mathsf{NP/N} & \mathsf{N} \\
\hline
\multicolumn{2}{c}{\mathsf{S_{pt}\backslash NP}\ >} & \multicolumn{2}{c}{\mathsf{NP}\ >}
\end{array}$$

Assuming that such cases can be detected and are corrected, it is not clear that their bottom-up translation procedure yields different results from our top-down method if the same heuristics are used to identify heads and distinguish between complements and adjuncts. In step 4, they assign variables to the lexical categories of words which have not been assigned categories yet, then traverse the whole tree bottom-up and instantiate these categories using the head/complement/adjunct information already available to instantiate these variables. However, some of this information will only be available at the top of the (sub)tree, and will thus presumably be percolated down the tree through the variables. In such cases, the resulting categories should be the same.

As Watkinson and Manandhar use AB categorial grammar, which only has function application (see §3), it is also not clear how they could extend their

[16] It is also not the correct analysis for NPs which only consist of one `DT` daughter, such as the following: `(NP (DT those))`, `(NP (DT some))`, `(NP (DT all))`

algorithm to deal with the `*T*` and `*RNR*` traces in the Treebank. Furthermore, they could not strip off the outer arguments of the head category when determining the categories of adjuncts (see §5.1), because AB categorial grammar does not allow composition. We would expect this to lead to a larger, less compact lexicon.

6 Supplementing the Acquired Lexicon

As discussed in §1, there have been other projects to build large CCG grammars automatically. These all share the property that their focus is on syntactic coverage with little or no attention to corresponding semantics. This is unfortunate since syntactic/semantic transparency is such an attractive aspect of the formalism. Hand-built CCG grammars can be easily designed so that every syntactic category is paired with a tailored semantic form, but it is very laborious to manually develop wide coverage grammars or lexicons. The automatically acquired lexicon helps by providing an extensive syntactic and lexical coverage along with frequency information for word/category pairs. However, an acquired lexicon such as that presented in §5 is deficient in that it lacks semantics and morphological information. Here, we propose a way in which an acquired lexicon can be augmented with semantics using the lexicon discussed in §4. This information can then be used in the knowledge components of Grok to maintain discourse information.

In Grok, various modules, such as the parser, query the lexicon module for the lexical entries of a particular word. That module, in turn, queries the hand-built and acquired lexicons. These two sets of results must then be combined. The ideal case is for the entries of the hand-built lexicon to mirror those of the acquired lexicon. Then, it is a simple matter to take the morphological features and the semantics from the hand-built lexicon to supplement the acquired one. Unfortunately, it is common for each lexicon to propose some categories that the other does not. This gives rise to three situations.

The first is the simple case where a syntactic category is proposed both by the hand-built lexicon and the acquired lexicon. In this case, the category is accepted. A small concern is that the categories are probably not exactly the same since the hand-built lexicon generally proposes significantly more morphological information. Thus, we unify the two syntactic categories and include the semantics from the hand-built category and the statistical information from the acquired category. For example, (65) shows compatible categories for **that** proposed in both lexicons. The hand-built syntactic category has more morphological information, and the acquired lexicon specifies that the extracted argument is the subject. The nondirectional slash variable '|' can unify with either a forward or backward slash. The unified category contains all of this information. The combined lexical entry also contains the semantics from the hand-built entry.

(65) a. Hand-Built
that $\vdash$ $(\mathsf{NP}_{\mathsf{bare}+}\backslash\mathsf{NP}_{\mathsf{bare}+})/(\mathsf{S}_{\mathsf{decl}}|\mathsf{NP})$: $\lambda p\lambda q\lambda x.p(x) \wedge q(x)$

b. Acquired
that $\vdash$ $(\mathsf{NP}\backslash\mathsf{NP})/(\mathsf{S}_{\mathsf{decl}}\backslash\mathsf{NP})$

c. Combined
that $\vdash$ $(\mathsf{NP}_{\mathsf{bare}+}\backslash\mathsf{NP}_{\mathsf{bare}+})/(\mathsf{S}_{\mathsf{decl}}\backslash\mathsf{NP})$: $\lambda p\lambda q\lambda x.p(x) \wedge q(x)$

The second case is where, for a particular word, a category is specified in the hand-built lexicon but not in the acquired lexicon. It is tempting to disregard such a category as being the result of the lexicon's over-generation: it will, for instance, propose an intransitive category for **devours**. However, it is possible that the category is valid but could not be acquired from the corpus. For example, we do not acquire categories for multi-word lexical items like **other than** and **such as**, which are given categories in the hand-built lexicon. Fortunately, these categories almost always occur for closed-class items. Thus, we employ the heuristic of taking the category if it is in a closed-class family and disregarding it otherwise.

Since the acquired lexicon will only contain entries which it has observed in its training data, it does contain gaps in its coverage of some open class words (see §5). In the event that the acquired lexicon contains *no* entries for a given word, we allow the word to take all of the categories which the hand-built lexicon can assign to it, thus favoring overgeneration to undergeneration.

The third and most difficult case is when a word has an acquired category not in the hand-built lexicon. This frequently occurs in large open classes where the word does not occur in the XTAG morphological database used by Grok. For example, the morphological database does not have an entry for **sneaker** while the acquired lexicon correctly proposes an NP category.

Lacking a corresponding entry in the hand-built lexicon results in impoverished morphology and no semantic information. One might expect that the tight connection of syntax and semantics inherent in CCG would allow one to deduce semantics directly from categories. Were it the case that each syntactic category had exactly one possible semantic category, this would indeed be trivial. But this is not the case, as shown by (66).

(66) a. **big** $\vdash$ NP/NP: $\lambda p\lambda y.p(y) \wedge \mathtt{big}(y)$

b. **dog** $\vdash$ NP/NP: $\lambda p\lambda y.p(y) \wedge rel(\mathtt{dog}, y)$

In this simple example, **big** and **dog** are both NP modifiers. However, while **big** compositionally combines with an NP, such as **house**, to denote an entity which is both a house and big, **dog house** does not denote an entity which is both a house and a dog. Instead, a different semantic relation, *rel*, is communicated. We do not attempt to determine what this relation is (it is different for

dog house, **dog sled**, **dog slobber**, etc.) since this is a difficult, well-studied problem that is not germane to this paper. Rather, the point is that adjectives and nouns have a common syntactic category in this lexicon but different semantics. This is one example of a general problem.

It seems obvious in this case that these examples can be distinguished by the fact that **dog**, in the Treebank, is annotated with the noun POS tag (`NN`) while **big** is annotated as an adjective (`JJ`). However, in general there are many possible features that can affect these choices—syntactic arity, syntactic features, morphology, etc. Therefore, rather than attempting to hand-code heuristics to determine semantic categories from syntactic categories, we have used a machine learning approach.

We started with randomly selected entries from the acquired lexicon while maintaining the distribution of word/category pairs observed in the corpus. We excluded numbers, punctuation, and proper names as these are handled in Grok through other means (see §7).

For each of these lexical entries, we produced training data consisting of the following features: the orthography of the word, its suffix, the number of syntactic arguments, various features of the acquired category, and whether the syntactic category is a verb, verb modifier, noun phrase, or noun phrase modifier. Also included is the correct semantic template for the lexical entry. These semantic forms include those for basic verbs, adjectives, pronouns, determiners, noun-noun constructions (e.g. **dog house**), prepositions, and alternative phrases. There are 21 semantic templates in total. These semantic forms contain several ambiguities like that in (66), but even so, it is still a rather impoverished set. It is sufficient, though, to fulfill the goals in practical applications such as discussed in [4], in which information from alternative set words such as **other** and **such as** is used to boost results for natural language search and information retrieval. We still lack information about the semantics for categories that do not appear in the hand-built lexicon, but, as discussed later in this section, these are not frequent.

We tested both the decision tree and maximum entropy approaches to machine learning on this task, using Ripper [9] to induce a decision tree, and our own maximum entropy software to build the maxent model (see §7 for more information).

Out of 10,000 entries selected at random from the acquired lexicon, 8866 were contained in the hand-built lexicon, and thus had associated semantics. Of these, 90% were used for training and 10% for testing.

As shown in Table 6, both the decision tree and the maximum entropy approaches perform in the ninety-seventh percentile at predicting the correct semantic form given the features described above. We compare these results to a baseline which simply chooses the most frequent semantic form for a word and guesses the overall most frequent semantic form if the word is unknown.

	training	testing
Baseline	86.4%	73.2%
Ripper	97.2%	97.3%
Maxent	97.8%	97.1%

TABLE 6. Evaluation of Inferring Semantics

As one can see, this performs significantly worse than both machine learning techniques.

To interpret the usefulness of these results, we present a few more statistics, illustrated in Figure 2. For 95.8% of the tokens in the corpus from which we acquired the lexicon, the syntactic category was available in the hand-built lexicon. For 88.7% of the tokens, the exact entry was found in the hand-built lexicon.

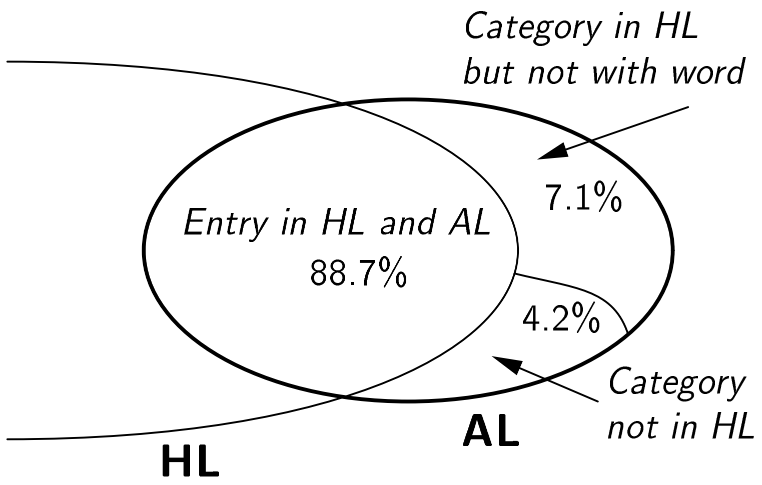


FIG. 2. Overlap of Acquired Lexicon and Hand-built Lexicon

Thus, for 7.1% of the tokens, the hand-built lexicon contains the appropriate category but does not associate that category with the lexical item. These are the cases for which we believe our models will provide good performance for guessing semantic templates. It is possible that these particular lexical items are fundamentally different from the other 88.7%, which is why their entries were not found in the hand-built lexicon. However, our morphological database does not contain a number of open class words (*e.g.* **sneaker**, as described above). Thus, it is exactly for these tokens that the approaches evaluated in Table 6 are meant. We conjecture that if the corpus from which the lexicon is acquired is representative, we can predict the correct semantic form for the 7% of the tokens which do not have a complete entry in our lexicon around 97% of the time.

The remaining 4% of the tokens are those for which the hand-built lexicon does not even contain the syntactic category. Since the training data only consists of entries contained in the hand-built lexicon, we have no reason to believe that the statistical models will perform well in predicting their semantic forms. It would be possible to test this by hand-building a test set from this class, but we have not done so for two reasons. First, many of these categories occur only once in the Treebank, and are partly the result of noise in the corpus. They can therefore be discounted. Second, the lexical entries produced when creating a test set could be easily integrated into the grammar and used to retrain the statistical models.

7 Preprocessing

We have discussed a robust way of automatically generating a large-coverage CCG lexicon from annotated corpora. Nonetheless, everyday text, such as that found in newspapers, still poses a number of problems for a CCG parser — even if it uses a lexicon which has been extracted from that very domain. In addition to obvious and universal problems such tokenization and sentence detection, problems more specific to a system such as Grok which uses a lexicalized grammar include subsets of the unknown word problem: e.g. recognizing dates and named entities, and dealing with abbreviated terms and numerical units in the text. These are textual elements for which the expressivity of a categorial analysis is unnecessary since pattern matching and statistical techniques do an excellent job of identifying many of them.

These are problems faced by all natural language understanding systems, and a great deal of research has gone into preprocessing techniques to deal with them. Our solution for these issues has been to create a pipeline of preprocessing components which use XML [38] documents in the input/output specifications. Following work by [26], we have built the components themselves using the maximum entropy framework. At present, we have built a tokenizer and sentence detector which use maximum entropy models with features based on those presented in [26] and [28]. Other tasks such as paragraph detection or dealing with figures and tables have not been implemented as yet, but the existing components have been designed to work with XML documents in a manner which facilitates the incorporation of such additions.

With tokenized and sentence detected data in hand, we are much closer to the goal of feeding the text to the CCG parser. However, there are still many things we can do to ease the parser's work and reduce the burden on the lexicon. One such task is named entity recognition. We feel that names should not be derived through standard CCG derivations for several reasons. First, the acquired lexicon is limited to those names it observes in the data. Furthermore, for a name such as **John**, the acquired lexicon produces two categories: NP for the name alone and NP/NP as in *John Smith*. This is particularly detrimental since a large factor in the efficiency of CCG parsing is dependent on lexical ambiguity and names are quite common in texts. Finally, names lack recursive structure, are not compositional, and fit a somewhat standard format, so finding a CCG derivation for them will not tell us much of interest. We thus choose to put the burden of deriving names on another component. Then, when a sentence is given to the parser, it receives names as chunks. A side benefit of this is that we can use the results of name detection to know whether or not we can decapitalize sentence initial words so that we do not need dual lexical entries for non-names occurring at the beginning of sentences. The results for evaluation on 20,468 training sentences and 4552 unseen testing sentences are given in Table 7. Named entity recognition is a well studied task, so it should be a simple matter to substitute a more developed approach such as [22], which achieves about 98.5% accuracy on unseen data and was the winning entry in MUC-7 [23].

	training	testing
precision	96.1%	95.9%
recall	96.9%	94.8%

TABLE 7. Evaluation of Name Detection

This approach can be extended to other phenomena such as dates, appositives, and compound nouns. Dates can be handled similarly to names. Both nominal appositives (such as *the Dutch publishing group*) and adjectival appositives (such as *61 years old*) have a very restricted syntactic positioning and are clearly delimited by commas, so we expect another maximum entropy component would do well in detecting them. However, unlike names, appositives have a compositional meaning and can have recursive structure. Even so, at the top they are still of the category NP or NP\NP. With appositives detected, we could give the parser the appositive's lexical items with the goal NP for nominal appositives or NP\NP for adjectival appositives in order to compute its meaning before proceeding with the rest of the sentence. Appositives also highlight the problem of commas, which receive a costly number of categories if no preprocessing steps such as these are taken.[17]

[17]Note that the current implementation of the lexicon acquisition algorithm assigns the pseudo-category

Cutting up the text in the manner described above can yield significant gains for a CCG parser. The following extract from the Wall Street Journal provides an excellent example of the benefits.

(67) *Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29. Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.*

If we were to perform name, date, and appositive detection on that text, a cubic chart parser would visit only 25% of the cells it would using the unprocessed tokens. Furthermore, since the complexity for lexicalized grammars such as CCG is also dependent on the number of categories per word, the reduction in the number of categories for dates, numbers, and names provided by dealing with them in this way will translate into yet more efficiency gains, especially since the maximum entropy components themselves are extremely efficient. We are also interested in using finite-state transducers for other tasks like date recognition and morphological analysis.

Finally, we have implemented a part-of-speech tagger based on [26] and have extended it to perform "CCG-tagging", similar to the supertagging of [29]. The model was trained using a CCG-annotated version of the Treebank (built by the process described in §5) as data. Initial model training on small training sets of 1,500 sentences gives accuracies of 82.4% per word accuracy on unseen data. For 12.3% of the sentences in the unseen test data, we achieve 100% per word accuracy. This compares with a baseline of 73.20% per word accuracy, if we always assign the most likely category to words occuring in the lexicon, and $\mathsf{NP}_{\mathsf{bare}+}$ otherwise. Using this baseline approach, only 1.54% of the sentences are tagged with 100% accuracy. Ultimately, this will serve as a component for improving category selection and thereby cutting down the parsing search space for most sentences.

The methodology of reducing complexity through preprocessing steps is generally applicable to the processing of rule-based formalisms. Since Grok is a module library, its components can be glued together to make systems with different capabilities and properties. For example, the preprocessing subsystem can be used completely independently of the parsing subsystem and could be used as a front-end to another system such as XTAG. The components are also language independent; for example, a new maximum entropy model for Portuguese POS-tagging could easily be used in place of the default English model.

"," to commas. In the Treebank, nominal appositives cannot be distinguished from noun phrase lists. Therefore, we treat them like the coordination/list case.

8 Future Work

In this paper, we have presented an algorithm for the acquisition of large categorial lexicons from the Penn Treebank, and we have shown how these syntactic lexicons can be supplemented with simple semantics. We have also shown how preprocessing steps are integrated in our system. However, there are a number of open issues, which we have not addressed yet.

A lexicon acquired in the manner described in this paper can only contain entries generated by instances occurring in the training corpus, but cannot generalize to sentences which require different category assignments. In our domain, there are three different kinds of generalizations we would like our system to be able to do.

The first generalization concerns the ability to deal with unknown words. Although we have seen in §5.12 that most unknown words are either NP or $\mathsf{NP}_{\mathsf{bare+}}/\mathsf{NP}_{\mathsf{bare+}}$, this is not true for all unknown words. One way in which we can deal with the problem of unknown words is to assume that the input to our system is part-of-speech tagged. We can use our program to determine the set of possible CCG categories for each part-of-speech tag. Then we can back off to the part-of-speech tag for words which do not occur in the lexicon. By doing this, we guarantee that we can assign categories to every word.

The second generalization regards other distributional regularities, corresponding to rules of the form "if a word has category A, then it also has category B". For instance, adjectives in English have the category $\mathsf{S}_{\mathsf{adj}}\backslash\mathsf{NP}$ when used predicatively, and $\mathsf{NP}_{\mathsf{bare+}}/\mathsf{NP}_{\mathsf{bare+}}$ when used attributively. Carpenter [6] gives a number of such lexical rules for categorial grammar, and we could just apply his rules to our lexicon. But as not all of his analyses correspond to the categories obtained from the Treebank, it is difficult to assess the impact these rules would have on the coverage of the lexicon.

The third generalization we would like our system to make concerns rules of derivational and inflectional morphology. In English, such rules are particularly important for verbs. Once we know the category of a present tense verb such as **love**, we also know the category for its third person singular form **loves**, and if we know that **loving** and **loved** are the present and past participles of **love**, we also know their categories. Such rules could be hand-coded, but we have not attempted to do this yet.

Chen and Vijay-Shanker [7] use the information encoded in the XTAG tree families to extend their extracted TAG lexicon to unobserved entries of (seen) words and (seen) elementary trees. They report that this leads to a 7.4% decrease of the missed <seen word, seen tree> cases on the test corpus (from 4.98% to 4.61%), whereas the lexicon more than triples in size. We would expect similar results on our data.

Another, equally important, issue is category selection during parsing. We have demonstrated that the lexicons extracted from the Treebank have wide coverage of unseen data, but the fact alone that we can find the desired category in the lexicon does not mean that we are guaranteed to actually use this category during parsing. Supertagging with CCG categories as described above is one approach towards category selection. In [16], we have taken another approach, using generative models over normal-form CCG derivations.

9 Conclusions

In this paper, we have shown how we provide additional coverage to a hand-built symbolic NLP system by using information gathered from an annotated corpus. We have automatically acquired a large syntactic CCG lexicon from the Penn Treebank, and we have shown how this lexicon can be combined with semantic information in the original system using machine learning techniques. We also have described how well-understood statistical preprocessing methods can be used and integrated in our system to improve the efficiency of parsing. We feel that these ideas will provide a general framework for expanding the capabilities of symbolic grammars so that they can be used in efficient real-world applications, countering claims that formalisms such as categorial grammar are impractical and difficult to scale.

10 Acknowledgements

We would like to thank our supervisors Mark Steedman and Bonnie Webber for their advice on and support of the work presented in this paper, and Chris Brew for advice on initial work on lexicon extraction from the Treebank. Geert-Jan Kruijff, Stephen Clark, Miles Osborne, Ambrose Nankivell, Ofir Zussman and three anonymous reviewers also provided many helpful comments and suggestions. Furthermore, we would like to thank Mary McGee Wood and other participants of the ESSLLI'2000 Workshop on Linguistic Theory and Grammar Implementation for stimulating discussions and feedback. We also gratefully acknowledge financial support from an EPSRC studentship, EPSRC grants GR/M96889 and GR/M75129, two Overseas Student Research awards, the Edinburgh Human Communication Research Centre/Language Technology Group, and the Division of Informatics.

References

[1] Anthony E. Ades and Mark J. Steedman. On the Order of Words. *Linguistics and Philosophy*, 4:517–558, 1982.

[2] Kazimierz Ajdukiewicz. Die Syntaktische Konnexität. In Storrs McCall, editor, *Polish Logic 1920-1939*. Oxford University Press, 207-231, 1935. Translated from *Studia Philosophica*, 1, 1-27.

[3] Yehoshua Bar-Hillel. A Quasi-Arithmetical Notation for Syntactic Description. *Lan-*

guage, 29:47–58, 1953.

[4] Gann Bierner. *Alternative Phrases: Theoretical Analysis and Practical Applications.* PhD thesis, Division of Informatics, University of Edinburgh, 2001.

[5] Joan W. Bresnan, Ronald M. Kaplan, Stanley Peters, and Annie Zaenen. Cross-Serial Dependencies in Dutch. *Linguistic Inquiry*, 13:613–636, 1982.

[6] Bob Carpenter. Categorial Grammars, Lexical Rules, and the English Predicative. In Levine, editor, *Formal Grammar: Theory and Implementation*, Oxford University Press, 1992.

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

[8] David Chiang. Statistical Parsing with an Automatically-Extracted Tree Adjoining Grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, October 2000.

[9] William Cohen. Fast Effective Rule Induction. In *Machine Learning: Proceedings of the Twelfth International Conference*, 1995.

[10] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing.* PhD thesis, University of Pennsylvania, 1999.

[11] Haskell B. Curry and Robert Feys. *Combinatory Logic: Vol I.* North Holland, Amsterdam, 1958.

[12] Christy Doran and Bangalore Srinivas. Developing a Wide-Coverage CCG System. In Anne Abeillé, Owen Rambow, editors, *Tree-Adjoining Grammars.* CSLI Publications, 2000.

[13] John Dowding, Jean Mark Gawron, Doug Appelt, Lynn Cherny, Robert Moore, and Douglas Moran. Gemini: A Natural Language System for Spoken Language Understanding. In *Proceedings of the 31st Annual Meeting of the Association of Computational Linguistics*, Columbus, OH, 1993.

[14] Free Software Foundation. GNU Lesser General Public License, 1991. http://www.gnu.org/copyleft/lesser.html.

[15] Claire Grover, John Carroll and Ted Briscoe. The Alvey Natural Language Tools Grammar. Technical report, The University of Edinburgh, Human Communication Research Centre, 1993.

[16] Julia Hockenmaier. Statistical Parsing for CCG with Simple Generative Models. In *Proceedings of the Student Research Workshop of the 39th Annual Meeting of the Association of Computational Linguistics and the 10th Conference of the European Chapter*, Toulouse, France, 2001.

[17] Aravind Joshi. Tree Adjoining Grammars. In David Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge, 1988.

[18] Aravind Joshi, K.Vijay-Shanker, and David Weir. The Convergence of Mildly Context-Sensitive Formalisms. In Peter Sells, Stuart Shieber, and Tom Wasow, editors, *Processing of Linguistic Structure*, pages 31–81. MIT Press, Cambridge MA, 1991.

[19] Ronald Kaplan and Joan Bresnan. Lexical-Functional Grammar: A formal system for grammatical representation. In *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA, 1982.

[20] David M. Magerman. Natural Language Parsing as Statistical Pattern Recognition. PhD Thesis, Stanford University, 1994.

[21] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330, 1993.

[22] Andrei Mikheev. A Knowledge-Free Method for Capitalized Word Disambiguation. In *Proceedings of the 37th Annual Meeting of the Association for Compuational Linguistics*, pages 159–166, 1999.

[23] Andrei Mikheev, Claire Grover, and Marc Moens. Description of the LTG system used for MUC-7. In *Proceedings of the 7th Message Understanding Conference (MUC-7)*, 1998.

[24] Carl Pollard and Ivan Sag. *Head Driven Phrase Structure Grammar*. CSLI/Chicago University Press, Chicago, 1994.

[25] Adwait Ratnaparkhi. A Simple Introduction to Maximum Entropy Models for Natural Language Processing. Technical Report IRCS-97-08, University of Pennsylvania, Institute for Research in Cognitive Science, 1997.

[26] Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, 1998.

[27] Manny Rayner, Beth Ann Hockey, and Frankie James. A Compact Architecture for Dialogue Management Based on Scripts and Meta-Outputs. In *Proceedings of Applied Natural Language Processing*, 2000.

[28] Jeff Reynar and Adwait Ratnaparkhi. A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proceedings of the International Conference on Spoken Language Processing*, pages 16–19, Washington D.C., 1997.

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

[30] Mark Steedman. Dependency and Coordination in the Grammar of Dutch and English. *Language*, 61:523–568, 1985.

[31] Mark Steedman. Combinatory Grammars and Parasitic Gaps. *Natural Language and Linguistic Theory*, 5:403–439, 1987.

[32] Mark Steedman. *Surface Structure and Interpretation*. MIT Press, Cambridge Mass., 1996. Linguistic Inquiry Monograph, 30.

[33] Mark Steedman. Information Structure and the Syntax-phonology Interface. *Linguistic Inquiry*, 34, 2000.

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

[35] K. Vijay-Shanker and David Weir. The Equivalence of Four Extensions of Context-free Grammar. *Mathematical Systems Theory*, 27:511–546, 1994.

[36] Aline Villavicencio. Building a Wide-Coverage Combinatory Categorial Grammar. Master's thesis, Cambridge, 1997.

[37] Watkinson, S. and Manandhar, S. (2001). Translating Treebank Annotation for Evaluation. In *Workshop on Evaluation for Language and Dialogue Systems, ACL/EACL*.

[38] World Wide Web Consortium. Extensible Markup Language (XML). Web Page, 1997. http://www.w3.org/XML/.

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

[40] XTAG-group. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-98-18, University of Pennsylvania, 1999.