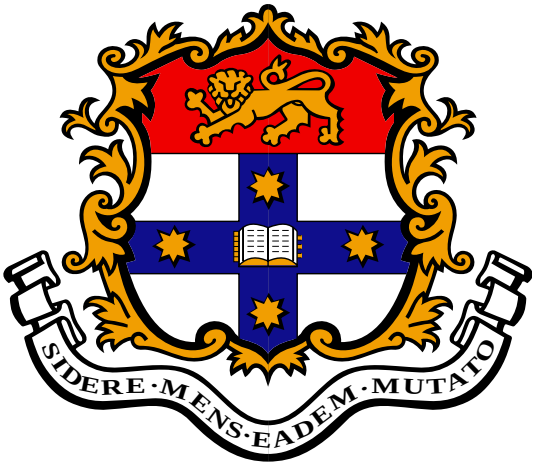
Wide-coverage efficient parsing with multi-modal combinatory categorial grammar

DANIEL GAR-SHON TSE

SID: 200415834



Supervisor: Dr. James Curran

This thesis is submitted in partial fulfillment of
the requirements for the degree of
Bachelor of Information Technology (Honours)

School of Information Technologies
The University of Sydney
Australia

5 November 2007

Acknowledgements

My gratitude is owed largely to three parties:

- My supervisor, James Curran, who in first year showed me that the intersection of computer science and linguistics was, to my great delight, not the empty set.
- My friends and colleagues, who provided light in-car entertainment and company for numerous meals throughout the Honours journey.
- My mother, father, brother and extended family, who together played a thousand under-acknowledged parts in the completion of this work.

Semper sint in flōre.

Statement of compliance

I certify that:

- I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure;
- I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);
- this Work is substantially my own, and to the extent that any part of this Work is not my own I have indicated that it is not my own by Acknowledging the Source of that part or those parts of the Work.

Name: *Daniel Gar-Shon Tse*

Signature: Date: *5 November 2007*

Contents

List of Figures

List of Tables

Abstract

Combinatory Categorial Grammar is a formalism for the specification of natural language syntax. It forms the theoretical foundation of Clark & Curran (C&C), an efficient, wide-coverage parser at the forefront of statistical parser research.

The goal of this work is to augment C&C for *multi-modal Combinatory Categorial Grammar* (MMCCG), a refinement of the CCG formalism which offers numerous theoretical and practical benefits. Prior to this work, no wide-coverage MMCCG corpus has ever been extracted or created, nor has MMCCG ever been realised in a wide-coverage parser, or evaluated on a large scale in the literature.

We present a methodology for the extraction of a large-scale training corpus. We develop and apply this method to automatically convert CCGbank, a corpus of 1.3 million words annotated for the CCG formalism, to support MMCCG.

Without a wide-coverage parser, we cannot validate many of the benefits claimed for MMCCG. To tackle this, we finally modify a state-of-the-art, wide-coverage CCG parser for MMCCG, enabling us to perform the first wide-coverage evaluation of MMCCG in the literature.

Wide-coverage parsing is a necessity for natural language processing applications which deal with the deep structure of text, such as many question answering systems. The benefits of efficiency and generality attributed to MMCCG will flow directly into improving applications built on MMCCG parsing.

CHAPTER 1

Introduction

Parsing is the recovery of hierarchical linguistic structure from linear text. The accurate and efficient recovery of this structure, which describes the relations which hold between the elements of a sentence, is a key step in any *natural language processing* (NLP) application which deals with deep semantic structure.

Although shallow heuristics for extracting these relations can account for them in limited situations, the full power of a natural language parser is required in order to capture the numerous ways in which the same semantics may be captured in different wordings.

(1.1) Hortense walked the dog today.

(1.2) It was today that Hortense walked the dog.

(1.3) Today, the dog was walked by Hortense.

(1.4) As for the dog, Hortense walked it today.

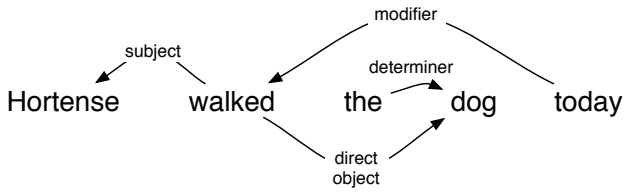


Figure 1.1: Dependency output from a natural language parser

The research we undertake deals broadly with a parsing formalism known as *Combinatory Categorial Grammar* (CCG), and in particular, the implementation and evaluation of an extension to CCG known as *multi-modal Combinatory Categorial Grammar* (MMCCG).

Typically, a parser is parameterised on a *lexicon*, a resource which maps words in the input language to a representation useful to the parser. The degree to which the lexicon by itself specifies a language is known as *lexicality*. The primary focus of this work is improving a CCG parser's lexicality: the degree to which the parser's lexicon defines a language. Lexicality is a desirable quality: it improves the usefulness of parsers – a single parser implementation will suffice to parse arbitrary languages. At the same time, it aids linguists who are compiling a lexicon in contributing their domain knowledge without requiring them to target a specific parser. Lastly, it can benefit the users of

any parser application – as we will show in our case, full lexicality grants efficiency benefits relative to a partially lexicalised CCG parser. We expect the efficiency of a parser to have an impact on the viability of natural language understanding systems, such as question answering systems, increasing their responsiveness for end users. In this work, we will also discuss the problems inherent in pure CCG which are addressed by MMCCG, while discussing the resources necessary to support MMCCG parsing and evaluating the theoretical and practical impact of MMCCG.

1.1 Combinatory Categorial Grammar

Combinatory Categorial Grammar is a highly lexicalised grammar formalism which unifies syntax and semantics through the act of parsing, by mapping directly between syntactic structure and the underlying logical (semantic) structure. CCG is capable of producing a semantic representation in parallel with the parsing act, since the operations which combine syntactic units are also capable of combining their functional representations. The ability to parse a query, and in doing so compositionally build the meaning of that query, gives it additional appeal for NLP applications such as automatic question answering [Bos, 2005].

The formalism itself is an application and extension of the framework of combinatory logic as established by Curry et al. [1958], in which *combinators*, higher-order functions, transform and compose other functions. CCG, as developed and described by Steedman [2000], interprets entries in a lexicon as types (*categories*), together with a set of combinators which act upon them to combine them into syntactic units. CCG itself is best understood as a small set of combinatory rules which combine adjacent categories into larger units.

The goal of any practical parsing formalism is to provide a means of encoding syntax (the assembly of words into larger units known as constituents), with the implicit goals of enabling the efficient, robust and accurate retrieval of syntactic and semantic structure from natural text.

1.2 Issues with Combinatory Categorial Grammar

In considering how to apply the pure formalism to parsing various structures found in the languages of the world, Steedman [2000] demonstrates that although we would otherwise like it to be, pure CCG alone is often insufficient to completely specify a language. In particular, in parsing a given language, it may not be desirable to allow the use of every combinatory rule given by the pure CCG formalism: as an example, one of the combinatory rules, namely *forward crossed composition*, is necessary to capture the free variation in the linear order of certain constituents possessed by languages such as Latin or Japanese. However, the rigidity of verb arguments in English, licensing

such a combinatory rule would clearly lead to *overgeneration*, that is, the acceptance of ungrammatical forms due to the unchecked application of CCG rules. This single point is one of the main motivations for multi-modal CCG [Baldridge, 2002], as introduced later. Steedman's solution to the overgeneration problem is to permit combinatory rules to be de-licensed for particular languages, or indeed particular structures. The problem with this approach is that it reduces the lexicality of CCG: since parsing with CCG now depends on information outside the lexicon, we lose some of the generality which makes CCG an attractive formalism.

A CCG parser may deal with the restricted applicability of combinatory rules by hard-coding these constraints into the implementation. This is unsatisfactory in two ways. Firstly, since the encoded restrictions are language-specific, extending the parser would require the modification and recompilation of the parser source code. Although the parser component should be language-invariant, CCG provides no way to encode applicability restrictions in the lexicon. In a practical implementation of a CCG parser, this means that restrictions must either be hard-coded into the parser, or else read in as a resource separate to the lexicon, in a parser-specific manner [Clark and Curran, 2007]. Secondly, embedding the restrictions in the parser incurs considerable overhead, as the parser must determine whether it is parsing an instance of a structure which disallows the application of a given combinatory rule. This must be done for each parsing step, and for each encoded exception. The promise of multi-modal CCG is to eliminate both shortcomings: the restrictions are moved out of the parser, so that the parser no longer has to consider where a rule *cannot* be used, as the categories in MMCCG now encode precisely which rules *can* be used.

Baldridge [2002] devised MMCCG as a solution to overgeneration that does not reduce lexicality. The key observation is that the rule restrictions necessary to prevent overgeneration should be encoded in the categories themselves, so that they can be encoded entirely in a lexicon. In MMCCG, each category, and hence each lexical item, specifies the set of combinatory rules in which it can validly participate. This category-driven derivational control prevents overgeneration without compromising on lexicality.

We evaluate not just the theoretical practicality of MMCCG, but its value as applied to wide-coverage parsing, by incorporating MMCCG into an existing parser capable of parsing general text.

Thesis 1. *Multi-Modal Combinatory Categorial Grammar yields the following benefits:*

- *increased generality in a parser through removal of hard-coded constraints*
- *category-driven, precise specification of syntax to prevent overgeneration*
- *reduced parser ambiguity and increased parser efficiency*

Since MMCCG returns the responsibility of specifying restrictions to the lexicon, the development of a lexicon which accurately captures those restrictions is vital. CCGbank

[Hockenmaier, 2003], a 1.3 million word English-language corpus, is a key resource enabling wide-coverage parsing in pure CCG. However, there is no corresponding resource for MMCCG.

In this work, we develop a technique for inducing a mode-annotated corpus, built upon CCGbank. With a MMCCG corpus in hand, we can quantitatively explore the claims made in Baldridge [2002] that MMCCG could yield benefits in parsing efficiency. Furthermore, the availability of a MMCCG corpus would fulfill a role analogous to the one CCGbank plays in CCG research, enabling future research to build upon the refinements introduced by MMCCG.

In the course of our work, we also perform an important transformation on CCGbank, to restore to it the quote symbols which the corpus derivation process of Hockenmaier [2003] strips away. The absence of quote symbols in the original CCGbank reduces its fidelity as a corpus and its usefulness in corpus applications including speaker segmentation, the goal of which is to identify alternations between tracts of direct speech and prose. We recover the quotes and produce a new CCGbank with quotes re-instated to its derivations.

We will also demonstrate that MMCCG greatly simplifies the internals of a CCG parser by shifting specificational responsibility out of the parser component, at the same time allowing for the integration linguistic domain knowledge into the parser without the need to modify parser internals.

Thesis 2. *It is possible to induce a* MMCCG *corpus by largely automatic means from an existing corpus.*

Thesis 3. *The theoretical benefits of* MMCCG *translate to practical benefits when implemented in a wide-coverage parser.*

Although Baldridge [2002] provides a proof-of-concept implementation of a MMCCG parser, it is targeted not for wide-coverage parsing, but as an experimental testbed for changes to the formalism. To date, the only evaluation which has been performed for MMCCG is Baldridge's informal evaluation by manual inspection on a hand-crafted test suite of 191 sentences. We implement MMCCG in a wide-coverage parser, so that for the first time, we can conduct a formal evaluation to validate the attractiveness of MMCCG over pure CCG.

Thesis 4. *A wide-coverage parser trained on an* MMCCG *corpus enables us to formally evaluate the benefits of* MMCCG *claimed by Baldridge [2002], and allows us to determine the impact of* MMCCG *relative to a pure* CCG *parser, and parsers based on comparable formalisms.*

1.3 Contributions of this work

Our first contribution is the very first application of MMCCG to a wide-coverage parser, the first time a wide-coverage MMCCG lexicon has been generated, and the first evaluation of a MMCCG parser on the wide-coverage parsing task.

The second contribution is the development of semi-automatic techniques for obtaining an MMCCG corpus, a prerequisite for our evaluation of the practicality of MMCCG parsing. Another product of our research will be a wide-coverage MMCCG corpus for the English language, derived from CCGbank [Hockenmaier, 2003].

Just as Hockenmaier's development of CCGbank enabled the successful application of CCG to wide-coverage parsing as investigated by Clark and Curran [2007], we hope that our development of a MMCCG corpus will be a key contribution to the advancement of CCG research in general, and a foundation for further research and interest in MMCCG.

1.4 Methodology

The fundamental advantage of MMCCG is its ability to provide more, and finer-grained information to the parser. Having applied semi-automatic corpus conversion techniques to the problem of acquiring a wide-coverage MMCCG corpus, the first evaluation task is an analysis of its impact on the parser subsystem which assigns categories to words, known as the *supertagger*. The performance of our derived corpora on the supertagger will give us an indication of the impact of multi-modal CCG on the parser system at large.

The second evaluation task is to determine the value of MMCCG as a general-purpose grammar formalism over pure CCG. We consider a number of attributes to be desirable in a wide-coverage parser:

- The ability to process text efficiently with respect to time and memory
- The ability to robustly provide accurate parses for a broad spectrum of text
- Ease of supporting new languages, while maintaining a clean division between the lexicon (grammar) and the parser

To evaluate the success of our MMCCG parser, we will employ a *dependency-based* metric as introduced by Clark and Hockenmaier [2002], computed from the number of dependencies (represented by the edges in Figure 1.1) in a candidate parse relative to a *gold standard* (data known to be correct).

We believe that performing our evaluation with respect to such a metric has several benefits: as argued in Eisner [1996], dependency-based metrics are a better determinant of correspondence to a gold standard for the CCG parsing problem than the PARSEVAL metric, an alternative popular in the literature. Furthermore, employing a dependency-based evaluation allows us to directly compare the values computed for the output of our modified parser against the evaluation performed for the unmodified C&C parser [Clark

and Curran, 2007].

1.5 Outline of this work

No pure CCG approach has successfully tackled overgeneration without compromising lexicality. The extra derivational control of MMCCG is necessary for solving overgeneration problems while returning full specificational control to the lexicon. MMCCG is firmly established in theory, but its claims of increased generality and parser efficiency cannot be adequately validated until we elevate MMCCG from a theoretical grammar formalism to the basis and foundation of a wide-coverage parser. Two key resources are required to achieve this goal: a wide-coverage corpus annotated for MMCCG, and a parser capable of enforcing the restraints specified by this MMCCG corpus. In this work, we achieve both of these for the first time in the parser literature: we develop a methodology for the semi-automatic derivation of such a wide-coverage MMCCG corpus, while retrofitting a state-of-the-art CCG parser to realise the benefits of MMCCG. Having both a MMCCG corpus and parser, we are finally able to perform an evaluation of the impact of MMCCG, fulfilling our original goal of validating the MMCCG as a practical grammar formalism.

A summary of this work is to appear in *Proceedings of the Australasian Language Technology Workshop (ALTW)*, to be held in December 2007, in Melbourne, Australia.

CHAPTER 2

Background

2.1 Formalisms and parsers

Combinatory Categorial Grammar [Steedman, 2000] is a powerful and concise lexicalised grammar formalism for the analysis of natural language. Lexicalised grammar formalisms, whose lexicons largely specify the syntax and semantics of a language, facilitate the fine-grained specification of syntactic behaviour, and the construction of extensible, language-independent parsers. LEXICALISED

Fundamentally, the role of a parser is to induce a hierarchical order from a linear sequence of tokens, such that meaningful units are identified, and coalesced into larger units. A simple model of language, well-known in both computer science and linguistics, is *context-free grammar*. To introduce a number of concepts in linguistics and natural language parsing, important to the remainder of our work, we will establish them in the familiar context-free grammar, before drawing parallels between this and the base formalism of the present work, Combinatory Categorial Grammar. CONTEXT-FREE GRAMMAR COMBINATORY CATEGORIAL GRAMMAR

A context-free grammar for a small subset of English is given in Figure 2.1a. The language of the grammar is defined as the set of strings which it generates, or equivalently, accepts.

S	$\rightarrow$	$NP\ VP\ .$
VP	$\rightarrow$	V_I
	|	$V_T\ NP$
	|	$V_{DT}\ NP\ NP$
NP	$\rightarrow$	$(Det)\ N$ | PN
N	$\rightarrow$	grapes | cake
V_T	$\rightarrow$	detests | devours
V_I	$\rightarrow$	sneezes | hiccups
V_{DT}	$\rightarrow$	gives | lends
Det	$\rightarrow$	the | a
PN	$\rightarrow$	Egbert | Hortense

(a) Productions for CFG C

Hortense devours the cake.
Egbert sneezes.
Hortense lends Egbert the grapes.

*Sneezes Hortense grapes the the
*Egbert sneezes grapes.
*The Egbert devours cake.

(b) Strings in and not in C

Figure 2.1: A CFG for a subset of English, and some strings from the language

Figure 2.1b shows a number of strings which are *not* in the language specified by the CFG fragment in Figure 2.1a[1]. Just as a CFG can be used to discriminate between strings which are in, or not in, the language it describes, an English speaker distinguishes ungrammatical forms from grammatical forms through a linguistic faculty known as *grammaticality judgement*. However, while CFG membership is a binary quality, the concept of grammaticality in linguistics admits shades of grey.

GRAMMATICALITY JUDGEMENT

Linguists have also traditionally analysed sentences by decomposing them into smaller meaningful units, known as *constituents*. Quirk et al. [1985] gives a syntactician's view of the part-whole structure of a sentence:

CONSTITUENT

> HIGHEST UNIT: SENTENCES, which consist of one or more
> CLAUSES, which consist of one or more
> PHRASES, which consist of one or more
> WORDS, which consist of one or more
> LOWEST UNIT: MORPHEMES
>
> (§2.7, pp 42–43)

The CFG fragment in Figure 2.1a captures some of the notions of language structure held by syntacticians: the start symbol S corresponds to the sentence, which is further decomposed into phrases (NP and VP, the noun and verb phrases), which in turn contain words (the terminals N, V, Det, PN of our grammar)[2].

The terminal labels, which correspond to classical parts of speech such as nouns, adverbs and adjectives, are also known as POS (part-of-speech) tags. Some parts of speech, such as the nouns and adjectives are called *open classes*, meaning that linguistic processes can generate new members of that class. For example, a steady stream of neologisms, such as the noun (and also verb!) senses of the words *blog* and *Google*, show that nouns and verbs are open classes. By contrast, other parts of speech such as the *pronouns* (including *they*, *she*, *I*) or *determiners* (such as *the*, *a*) are said to be *closed*, since new members are added extremely uncommonly. If there is a regularly applicable derivational process (known as a *productive* process) which converts a member of the open class X to a class Y (such as the gerund suffix *-ing* which converts a verb to a verbal noun, e.g. *The running of the bulls*), then Y must also be an open class.

POS TAG

OPEN CLASS

CLOSED CLASS

PRODUCTIVITY

The multiple productions for the CFG non-terminal VP models the concept of *subcategorisation frames*, which capture the variation in the number and kind of arguments taken by the verb. The three productions for VP correspond respectively to the subcategorisation frames for *intransitive* (one subject argument), *transitive* (one subject and one direct object), and *ditransitive* (one subject and two direct objects) verbs.

SUBCATEGORISATION FRAME

INTRANSITIVE
TRANSITIVE
DITRANSITIVE VERB

[1] Throughout this work, we use two conventions drawn from linguistics: we prefix ungrammatical forms with asterisks, while marginally grammatical/ungrammatical forms are prefixed with question marks.

[2] Our simple grammar does not capture *clauses*, which arise in compound sentences, nor does it model *morphemes*, since we treat the terminals as atomic.

Obligatory constituents are known as *arguments* or *complements*, while optional constituents are referred to as *adjuncts*. Unlike a complement, an adjunct may be deleted without affecting grammaticality. ARGUMENT/COMPLEMENT ADJUNCT

(2.1) Egbert lent Hortense a pony.

(2.2) *Egbert lent.

(2.3) Chocolate croissants lay on the table at the French-themed party.

(2.4) Chocolate croissants lay on the table.

The production $NP \rightarrow (Det)\ N$ also captures the fact that the class of determiners Det, although allowed in some circumstances (before a singular common noun such as *ball*), may be disallowed in others (preceding a non-count noun such as *water*).

(2.5) I kick *the* car.

(2.6) ?The water falls from the sky as rain.

Context-free grammar is an example of a *grammar formalism*, a formal system by which we can specify the syntax of a language. The role of a formalism is entirely descriptive: a method of representing a language does not specify how to parse a given string in that language. This role is held by the *parser*, which accepts a specification of the language, and decides membership, often with the side effect of generating the intermediate structure of a *derivation*. Well-known parsing algorithms include the shift-reduce algorithm for the language class *LR*, and the Cocke-Kasami-Younger (CKY) and Earley algorithms for the general context-free languages [Manning and Schütze, 1999]. GRAMMAR FORMALISM PARSER DERIVATION

Many of the linguistic concepts we have just introduced through the lens of the context-free grammar are reflected in CCG. With this in mind, we introduce the fundamental concepts which we will be using throughout this work.

2.2 Introducing Combinatory Categorial Grammar

CCG achieves lexicality by encoding its syntax in a resource known as the *lexicon*, which in CCG is a one-to-many mapping from a lexical item to objects known as *categories*, which completely and compactly define the interaction of that lexical item with other constituents. The formalism of CCG defines two things:

LEXICON
CATEGORIES

- the structure of categories
- a small set of *combinatory rules*, which combine categories together

COMBINATORY RULES

2.2.1 Structure of categories

Steedman [2000] specifies the structure of categories in the following way:

ATOMIC
COMPOUND CATEGORIES

DEFINITION 1. *Given a finite set of atomic categories $\mathcal{F}$, the set of categories $\mathcal{C}$ is the smallest set such that:*

- $\mathcal{F} \subseteq \mathcal{C}$
- $X/Y, X\backslash Y \in \mathcal{C}$ *if* $X, Y \in \mathcal{C}$

For example, if $\mathcal{F} = \{S,\ NP\}$, then examples of elements of $\mathcal{C}$ are S, $(S\backslash NP)/NP$ and $(NP\backslash NP)\backslash(NP\backslash NP)$. In any compound category $X \mid Y$, where $\mid$ stands for one of the directionality slashes $\{/,\ \backslash\}$, we call Y the *argument* category, and X the *result*. In this work, we informally refer to a *slash* interchangeably with the compound category in which it is contained.

ARGUMENT
RESULT
SLASH

We now outline two informal conventions associated with the structure of categories, which we will use throughout this work.

CONVENTION 1. ***Directionless slash.*** *When we are describing a compound category without regard to the directionality of its slash, we will informally write $X|Y$ to mean either of $X\backslash Y$ or X/Y.*

CONVENTION 2. ***Ordering on slashes.*** *We assign each slash an index, corresponding to a pre-order traversal on the category tree naturally induced by a category.*

Later on, when we need to speak about a particular slash (corresponding to a particular compound category), we will find it useful to impose an arbitrary indexing on them. For example, for the category $((A/B)/(C/D))\backslash(E/F)$, we have the category tree in Figure 2.2, and hence the indexing $((A/^2B)/^1(C/^3D))\backslash^0(E/^4F)$ on its slashes. We may call the slash of a compound category with index 0 the *topmost* or slash.

TOPMOST SLASH

We write the assignment of a lexical item w to category C as $w \vdash C$. A lexical item may be mapped to more than one category, resulting in *categorial ambiguity*. It is easy to see that categorial ambiguity is necessary to capture *polysemy* and *homonymy*. The word *can* exhibits polysemy between Examples 2.7 and 2.8, because the senses

CATEGORIAL AMBIGUITY
POLYSEMY
HOMONYMY

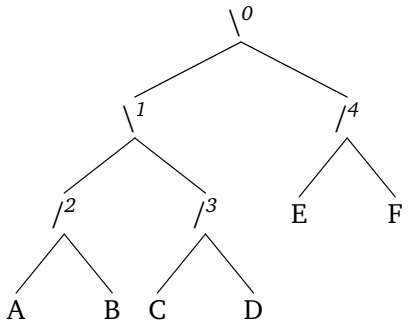


Figure 2.2: Category tree for $((A/B)/(C/D))\backslash(E/F)$

are semantically related via semantic extension, yet syntactically distinct. On the other hand, the word *can* in Examples 2.7 and 2.9 is said to exhibit homonymy, since the two exhibited senses of *can* are not semantically related.

(2.7) We can the tuna the day it's caught.

(2.8) Peaches come in a can.

(2.9) We can buy the cats later.

Each of the senses of *can* shown above corresponds to one of the below categories.

$$\begin{array}{lcl} \textit{can} & \vdash & (S[dcl]\backslash NP)/NP \\ \textit{can} & \vdash & N \\ \textit{can} & \vdash & (S[dcl]\backslash NP)/(S[b]\backslash NP) \end{array}$$

Although categories describe the way in which lexical items interact with other constituents, to induce structure in language we need to introduce combinatory rules, which form new categories from adjacent categories.

2.2.2 Combinatory rules

Categories are combined via a set of *combinatory rules*, which are an invariant part of the formalism. The combinatory rules are not part of the *lexicon*, which is solely the mapping from lexical items to categories. The complete set of CCG combinatory rules with the symbol representing each rule is given in Figure 2.3. Each rule is named by: COMBINATORY RULES

- a combinator: *application*, *composition* (**B**), *type-raising* (**T**), *substitution* (**S**)
- a directionality: *forward* (>), or *backward* (<)
- for combinators **B** and **S**, a variant: *harmonic* (unmarked) or *crossed* ($_\times$)

The categories on the left and right side of a combinatory rule are known as *input* and *output categories* respectively. A rule is either unary or binary, depending on whether INPUT/OUTPUT CATEGORY UNARY/BINARY RULE

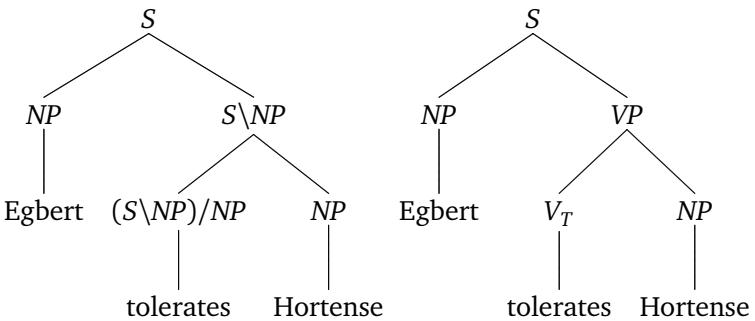
$$X/Y \quad Y \quad \Longrightarrow \quad X \quad (>) \tag{2.10}$$
$$Y \quad X\backslash Y \quad \Longrightarrow \quad X \quad (<) \tag{2.11}$$
$$X/Y \quad Y/Z \quad \Longrightarrow \quad X/Z \quad (>\mathbf{B}) \tag{2.12}$$
$$Y\backslash Z \quad X\backslash Y \quad \Longrightarrow \quad X\backslash Z \quad (<\mathbf{B}) \tag{2.13}$$
$$X/Y \quad Y\backslash Z \quad \Longrightarrow \quad X\backslash Z \quad (>\mathbf{B}_\times) \tag{2.14}$$
$$Y/Z \quad X\backslash Y \quad \Longrightarrow \quad X/Z \quad (<\mathbf{B}_\times) \tag{2.15}$$
$$X \quad \Longrightarrow \quad T/(T\backslash X) \quad (>\mathbf{T}) \tag{2.16}$$
$$X \quad \Longrightarrow \quad T\backslash(T/X) \quad (<\mathbf{T}) \tag{2.17}$$
$$(X/Y)/Z \quad Y/Z \quad \Longrightarrow \quad X/Z \quad (>\mathbf{S}) \tag{2.18}$$
$$Y/Z \quad (X\backslash Y)/Z \quad \Longrightarrow \quad X/Z \quad (<\mathbf{S}) \tag{2.19}$$
$$(X/Y)\backslash Z \quad Y\backslash Z \quad \Longrightarrow \quad X\backslash Z \quad (>\mathbf{S}_\times) \tag{2.20}$$
$$Y/Z \quad (X\backslash Y)/Z \quad \Longrightarrow \quad X/Z \quad (<\mathbf{S}_\times) \tag{2.21}$$

Figure 2.3: CCG combinatory rules

it takes one or two input categories. The only unary combinator in the above set of combinatory rules is type-raising (**T**).

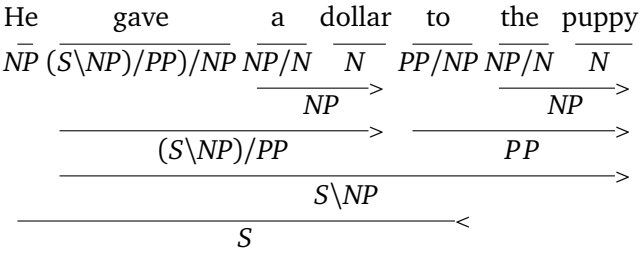
Rules 2.10 and 2.11 alone constitute a formalism known as the AB calculus, after Ajdukiewicz [1935] and Bar-Hillel [1953]. Bar-Hillel et al. proved in 1960 that the generative power (see Section 2.3) of the AB calculus is equivalent to that of the context-free languages. Figure 2.4 demonstrates that an AB calculus derivation is essentially a relabelling of a CFG derivation tree, where the category labels yield more useful information than the equivalent non-terminal labels in the CFG derivation.

AB CALCULUS

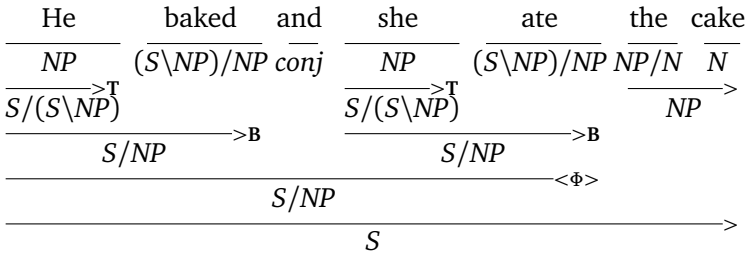


(a) AB derivation (b) CFG derivation

Figure 2.4: An AB-CFG equivalence



(a) Analysis of complement *PP*



(b) Complex coordination

Figure 2.5: CCG analyses

Two CCG derivations which use a number of the combinatory rules are depicted in Figure 2.5. The first example uses only the two rules of application to analyse a simple ditransitive verb sentence, while more powerful combinatory rules are employed in Example 2.5b to form a natural analysis of the syntax of *argument cluster coordination*, which we consider further in Section 2.4.3.

The limitations of context-free grammar are well known; a widely known result is that a language describing the syntax of *crossing dependencies* in Swiss German is outside of the context-free languages [Shieber, 1985], as is an analogous construction in Dutch, as shown in Figure 2.6 [Steedman, 2000].

CROSSING DEPENDENCIES

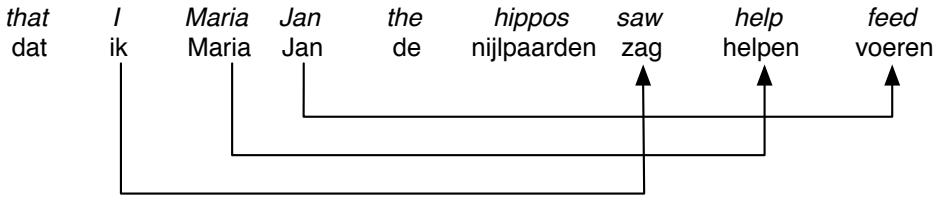


Figure 2.6: Crossing dependencies in Dutch: *that I saw Maria help Jan feed the hippos*

As for our own concern of parsing English with CCG, Baldridge and Kruijff [2004] shows that restricting ourselves to the AB calculus would result in an increase in the

number of potential categories, resulting in increased *categorial ambiguity*, which arises when the set of categories a lexical item is mapped to is large. Providing the full set of expressive combinators permits us to capture some syntactic constructions past the context free languages while remaining efficiently parseable, as well as allowing a language to be captured succinctly in fewer categories.

CATEGORIAL AMBIGUITY

The *generative power*, or expressiveness of a language is determined by the set of strings it can generate. A set of formalisms are said to be *weakly equivalent* if they are able to generate the same set of strings. A weak equivalence between the generative power of CCG and that of formalisms such as LIG (Linear Indexed Grammar [Gazdar, 1985]) and LTAG (Lexicalised Tree Adjoining Grammar [Schabes et al., 1988]) allows us to characterise the generative power of CCG as *mildly context-sensitive*, which renders it powerful enough to capture a wide range of natural language constructs, while still admitting polynomial time parsing algorithms [Weir and Joshi, 1988, Vijay-Shanker and Weir, 1993, Steedman, 2000].

GENERATIVE POWER

WEAK EQUIVALENCE

LIG

LTAG

MILD CONTEXT-SENSITIVITY

In English, the most productive of the combinatory rules are application ($>$, $<$), forward and backward harmonic composition ($>\mathbf{B}$, $<\mathbf{B}$), forward type-raising ($>\mathbf{T}$) and backward crossed composition ($<\mathbf{B}_\times$). As we will show in Section 4.1, forward crossed composition ($>\mathbf{B}_\times$) induces a freedom of word order which is undesirable in non-free word order languages such as English.

Each combinatory rule has a functional reflex, indeed the functional forms of these combinators are identical to those defined in combinatory logic [Curry et al., 1958]. This correspondence between syntactic and semantic structure gives CCG additional appeal for use in natural language understanding systems which require a model of semantic structure, since such a structure is constructed in tandem with the parsing process.

In this work, we will depict CCG analyses in three equivalent ways: firstly, with the application of combinatory rules made explicit, as in Figure 2.5 above. Secondly, we may represent a derivation in tree form (see Figure 3.13a), essentially the first representation in top-down, instead of bottom-up form. In this representation, the combinatory rule applications are implicit. Third, we can annotate the text of a derivation, as in Example 2.22, with a constituent grouped by square brackets followed by its category.

(2.22) $[[\text{He cooks}]_S \text{ and } [\text{he cleans}]_S]_S$.

2.2.3 Features

C&C [Clark and Curran, 2007] is a state-of-the-art *wide-coverage* parser based on CCG, to be described in much greater detail in following sections. Wide-coverage parsers are capable of parsing general text, as opposed to a small or restricted subset of a language. C&C acquires its statistical model with a wide-coverage corpus known as CCGbank [Hockenmaier, 2003], which we will enhance through our work of Chapter 3.

WIDE-COVERAGE

In C&C, categories optionally carry *features*, which in CCGbank are simply a set of annotations which yield more specific information as to the semantic or syntactic role of the constituent they represent. A table of all features attested in CCGbank is given in Table 2.1. Features allow an amount of semantic role information to be encoded in a category, while still allowing a bare (featureless) category to unify properly with any feature-laden category. For example, the category assignment *ask* $\vdash (S[dcl]\backslash NP)/S[qem]$ specifies that *ask* takes an embedded question argument, and not, for example, a plain sentence S:

FEATURE

(2.23) I asked [what the time was]$_{S[qem]}$.

(2.24) *I asked [the time is six]$_{S}$.

However, a feature-laden category matches when a featureless category is sought: for example, the below combinatory rule is a valid instance of backward crossed composition, because the featureless sought category $S\backslash NP$ is satisfied by the feature-laden category $S[dcl]\backslash NP$. However, due to the unification procedure, the result category gains the feature $[dcl]$.

UNIFICATION

$$(S[dcl]\backslash NP)/NP \quad (S\backslash NP)\backslash(S\backslash NP) \quad \Rightarrow \quad (S[dcl]\backslash NP)/NP$$

Purely from the form of the backward crossed composition ($<\mathbf{B}_{\times}$) rule given in Figure 2.3, we would expect the result to be $(S\backslash NP)/NP$. Strictly speaking, the category $(S\backslash NP)\backslash(S\backslash NP)$ carries a *feature variable*, which we may write as $(S[X]\backslash NP)\backslash(S[X]\backslash NP)$. According to the rule of backward crossed composition,

$$Y/Z \quad X\backslash Y \qquad \Longrightarrow \quad X/Z \quad (<\mathbf{B}_{\times})$$

the category $S[dcl]\backslash NP$ from the first input category unifies with $S[X]\backslash NP$ of the second. The unifier in this case is $X \leftarrow dcl$, and the assignment propagates to yield the category $(S[dcl]\backslash NP)\backslash(S[dcl]\backslash NP)$. Only then is the result category formed: $(S[dcl]\backslash NP)/NP$. In this work, we will represent features only when pertinent, to reduce the clutter in category representations.

Features allow us to specify further refinements of the category labels, while unification ensures that firstly, feature-laden categories can still combine with bare categories, and secondly, that feature information can be passed from the input categories to the output category in a controlled way. Without unification, it is not possible to replace feature instantiations with distinct categories (for example, *S-DCL* for $S[dcl]$), since they would then fail to match with a bare category such as S.

Features on any category

[*conj*]	Partially conjoined category (see Section 2.4.3)	

Features on *N* or *NP*

[*expl*]	Expletive *it*	*It is snowing today*
[*nb*]	Non-bare *NP* (one with a determiner)	*the white grapes*
[*num*]	Numeric *NP*	*three*
[*thr*]	Expletive *there*	*There are four lights*

Features on *S*

Clause features

[*as*]	*as* clause	*as though he was the boss*
[*adj*]	Adjectival complement	*aggressive in their actions*
[*asup*]	Superlative adjective clause	it was at *best problematic*
[*b*]	Bare infinitive, subjunctive, imperative clause	went to *water the garden*
[*bem*]	Embedded bare subordinate clause	that *he be sent to Portugal*
[*em*]	Embedded subordinate clause	I estimate *that it is three pounds*
[*for*]	*for* clause	It is hard *for me to read*
[*frg*]	Sentence fragment	What *about the alpaca?*
[*inv*]	Subject-verb inverted clause	So *did the culprit*
[*ng*]	*-ing* (participial) clause	He is *baking the pie*
[*pss*]	Passive clause	he is *rescued by a passing ship*
[*pt*]	Past participle clause	they had *gone to the cinema*
[*qem*]	Embedded question clause	I asked *what it was for*
[*to*]	*to*-infinitive	I asked *to be sent abroad*

Sentence features

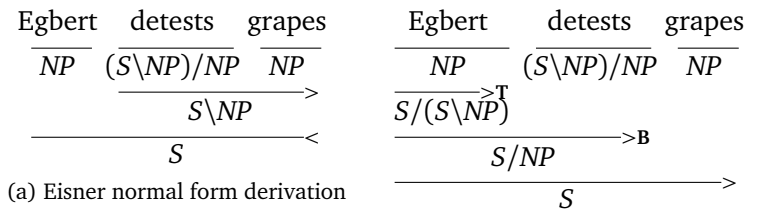
[*dcl*]	Declarative sentence	*He freezes the grapes*
[*intj*]	Interjection	*Uh oh*! *Quack*!
[*poss*]	Possible/contrary-to-fact sentence	as *if he was in the kitchen*
[*q*]	*Yes-no* question	*Did you examine the azaleas?*
[*wq*]	*wh*-word question	*What is its airspeed*?

Table 2.1: All features seen in CCGbank

2.2.4 Eisner normal form

The derivational freedom afforded by the power of combinatory rules has an impact on the efficiency of the parsing task. Consider the two derivations in Figure 2.7. We can readily see that the same category *S* spans both parses, although the sequence of combinatory rule applications differ. Although we do not show it here, the semantics induced by the functional analogues of the combinatory rules acting on the logical form of lexical items are identical in both cases.

This property of CCG, by which a set of derivations are semantically identical, but



(a) Eisner normal form derivation

(b) Non-Eisner normal form derivation

Figure 2.7: Eisner normal form

derivationally distinct, is known as *spurious ambiguity*. Eisner [1996] developed a constraint on derivations known as *normal form*. The key insight of Eisner normal form is twofold. Firstly, once a derivation in progress fails to satisfy the two Eisner normal form constraints (given below), we can discard that derivation. Secondly, it guarantees that for every equivalence class of parses which are semantically identical but derivationally distinct, there exists a representative parse which is in Eisner normal form, and hence is guaranteed to be found if the parser is restricted to considering normal form derivations.

SPURIOUS AMBIGUITY

EISNER NORMAL FORM

The two normal form constraints are defined by Eisner in terms of generalised composition, which we have omitted from the list of combinatory rules given in Figure 2.3. For simplicity, however, we will consider the case $k = 0$ of the general formulation, which is given in terms of the already-introduced combinators of application ($>$, $<$) and regular composition ($\mathbf{B}$).

DEFINITION 2 (Eisner normal form, $k = 0$). *A derivation is in* Eisner normal form *when both of the following conditions hold:*

- *No category produced by* forward *composition* ($>\mathbf{B}, >\mathbf{B}_\times$) *is ever used as a* left *argument to application or composition.*
- *No category produced by* backward *composition* ($<\mathbf{B}, <\mathbf{B}_\times$) *is ever used as a* right *argument to application or composition.*

While Example 2.7a is in Eisner normal form because composition is never employed, Example 2.7b is not, since composition is applied over *Egbert detests*, which then serves as the left argument to an application with *grapes*. Where spurious ambiguity arises, Eisner normal form guarantees that there exists a unique representative of the set of semantically identical derivations, which satisfies the two constraints of Definition 2.

As seen above, even the simplest declarative sentence involving a transitive verb exhibits spurious ambiguity. The insight of Eisner normal form greatly increases the

efficiency of CCG parsing, increasing its suitability in many applications.

2.3 The C&C parser

The practical focus of our work will be the C&C parser [Clark and Curran, 2007], a state-of-the-art wide-coverage statistical CCG parser. This system consists of three conceptual components:

- The *supertagger*, responsible for assigning a category to each lexical item. The C&C supertagger incorporates a maximum entropy classifier, using a set of *contextual predicates* (features) to capture the local context of a lexical item. The classifier uses this contextual information to inform its assignment of a category [Clark and Curran, 2004]. SUPERTAGGER
- The *parser*[3], which uses a form of the CKY algorithm to combine categories using the combinatory rules. The C&C parser uses a data structure known as the *chart* to efficiently store all candidate parses for a given string. PARSER CHART
- A parse scoring component, which assigns a probability to each candidate derivation according to a log-linear model, allowing us to select the most likely parse.

The close interaction of the supertagger with the parser enables efficient parsing: initially, the supertagger generates few categories, only increasing the number of candidates when no parse succeeds. A derivation whose top-level category spans the input tokens is known as a *spanning analysis*.

Making available the entire gamut of CCG combinatory rules in a parser can rapidly lead to *overgeneration*, the undesired acceptance of incorrect sentences. That is, in most languages, the full generative power of CCG is unnecessary, and in fact undesirable. In light of this, Steedman [2000] specifies that any given CCG grammar is free to specify the non-applicability of any of the combinatory rules, optionally where the input categories satisfy some structural condition. However, since CCG itself provides no means for specifying these restrictions, they must be encoded in an *ad hoc* manner. Since these restrictions are specified separately to the lexicon, this deficiency reduces the lexicality achievable by CCG. These restrictions are inefficiently hard-coded into the C&C parser, accordingly the parser contains numerous explicit checks to determine if argument categories to a combinatory rule satisfy structural constraints. This also is a major impediment to the generality of C&C, since these hard-coded restrictions only encode the rules of English syntax. The ultimate goal is to remove these restrictions by providing equivalent derivational restrictions in the lexicon, freeing the parser from performing these language-specific checks. OVERGENERATION

[3]We will use the term *"parser component"* as distinct from "parser" when we need to distinguish between the parsing subsystem, and C&C taken as a whole.

2.4 Multi-modal Combinatory Categorial Grammar

As a solution to overgeneration which does not reduce the lexicality of the formalism, Baldridge [2002] devised *multi-modal* CCG, where each slash of each category encodes an indication (or *mode*) defining the rules in which it may participate. We will describe the structure and interpretation of modes before describing the benefits they yield.

MULTI-MODAL CCG

MODE

2.4.1 Multi-modal CCG categories

Baldridge [2002] defines seven modes, arranged as a directed graph. An arc from mode s to mode t implies that mode t is licensed for at least every rule licensed by s. Formally, the set of MMCCG categories is defined by Baldridge [2002] in the following way.

DEFINITION 3. *Given a finite set of atomic categories $\mathcal{F}$, the set of* MMCCG *categories $\mathcal{C}$ is the smallest set such that:*

- $\mathcal{F} \subseteq \mathcal{C}$
- $X/_i Y, X\backslash_i Y \in \mathcal{C}$ *if* $i \in \{\star, \diamond, \diamond\!\!\times, \times\!\!\diamond, \triangleleft, \triangleright, \cdot\}$, $X, Y \in \mathcal{C}$

Intuitively, a MMCCG category is distinguished from a CCG category by the presence of a *mode* on every slash of every compound category. In this work, we will call the bare CCG category underlying a MMCCG category its *structural category*. For example, the structural category of the MMCCG category $(NP\backslash_\star NP)/_\triangleleft NP$ is the CCG category $(NP\backslash NP)/NP$.

STRUCTURAL CATEGORY

We will now see how these modes interact with the applicability of combinatory rules.

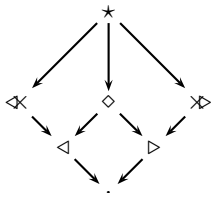


Figure 2.8: Hierarchy of modes

2.4.2 Combinatory rules in multi-modal CCG

We introduce each of the modes in Baldridge's mode scheme as depicted in Figure 2.8, from the mode which permits the fewest combinatory rules ($\star$) to the mode permitting all of them ($\cdot$).

The topmost tier contains the most restrictive *application-only* mode ($\star$), which only permits the two application rules ($>$, $<$). It is simple to give mode-aware versions of the

CCG application rules:

$$X/_\star Y \quad Y \quad \Rightarrow \quad X \qquad (>) \qquad (2.25)$$

$$Y \quad X\backslash_\star Y \quad \Rightarrow \quad X \qquad (<) \qquad (2.26)$$

These rules stipulate that for a category to serve as input to application, it must either carry the mode $\star$, or by the inheritance condition, be any super-mode of $\star$ in the hierarchy of Figure 2.8.

The second tier contains a mode $\diamond$ which permits harmonic composition, together with the application rules inherited from mode $\star$. Similarly, the modes $\nlessdot$ and $\ngtrdot$ in the same tier permit crossed composition. The motivation for the grammatical control governed by the directionality portion of these modes ($\triangleleft$ or $\triangleright$) is subtle, and we defer the explanation to Section 4.2.1. We will conflate $\nlessdot$ and $\ngtrdot$ into a single mode $\times$ when giving the mode-aware rules below:

$$X/_\diamond Y \quad Y/_\diamond Z \quad \Rightarrow \quad X/_\diamond Z \qquad (>\mathbf{B}) \qquad (2.27)$$

$$Y\backslash_\diamond Z \quad X\backslash_\diamond Y \quad \Rightarrow \quad X\backslash_\diamond Z \qquad (<\mathbf{B}) \qquad (2.28)$$

$$X/_\times Y \quad Y\backslash_\times Z \quad \Rightarrow \quad X\backslash_\times Z \qquad (>\mathbf{B}_\times) \qquad (2.29)$$

$$Y/_\times Z \quad X\backslash_\times Y \quad \Rightarrow \quad X/_\times Z \qquad (<\mathbf{B}_\times) \qquad (2.30)$$

The third tier contains the modes $\triangleleft$ and $\triangleright$, which permit *both* harmonic and crossed composition, while the bottom-most tier contains the maximally permissive mode $\cdot$.

Finally, the type-raising combinator (Rules 2.16 and 2.17 of Figure 2.3), which introduces two slashes, is slightly modified in multi-modal CCG:

$$X \quad \Longrightarrow \quad T/_i(T\backslash_i X) \qquad (>\mathbf{T}) \qquad (2.31)$$

$$X \quad \Longrightarrow \quad T\backslash_i(T/_i X) \qquad (<\mathbf{T}) \qquad (2.32)$$

The index i can be thought of as a "mode variable": in other words, the two slashes of a type-raised category may carry *any* mode, as long as they are the *same* mode.

Baldridge's goal for MMCCG is to provide a handle on the full power of CCG, allowing us to harness all of it when truly necessary, while limiting its availability when it is not. By giving a lexicon writer additional derivational control, MMCCG achieves benefits which flow directly from the underlying theory to their implementation in a parser. We describe a number of unsatisfactory CCG approaches to the analysis of a fundamental natural language construct, before describing a solution enabled by MMCCG.

2.4.3 Addressing overgeneration using MMCCG

An implicit goal of natural language parsing is to accept the grammatical, while rejecting the ungrammatical. A parser which fails to parse grammatical sentences is undesirable, and a parser which readily accepts ungrammatical sentences may be undesirable,

depending on the application.

The syntax of *coordination* conjoins multiple like constituents into a single constituent, which has the same syntactic role as its conjuncts. In English, the class of conjunctions includes *and* and *or*, which are interposed between each pair of conjuncts. coordination

The ease with which CCG handles coordination is one of its strengths [Hockenmaier and Steedman, 2001]: coordination between two constituents is possible precisely when they have the same category. With the type-raising and composition combinators, more complex cases such as Example 2.34 also yield natural analyses through CCG.

(2.33) The $[[\text{third}]_{N/N}$ and $[\text{fourth}]_{N/N}]_{N/N}$ doors lead to the $[[\text{kitchens}]_N$ and $[\text{dining room}]_N]_N$.

(2.34) $[[\text{I}_{S/(S\backslash NP)}\ \text{peeled}_{(S\backslash NP)/NP}]_{S/NP}$ and $[\text{he}_{S/(S\backslash NP)}\ \text{cooked}_{(S\backslash NP)/NP}]_{S/NP}]_{S/NP}$ $[\text{the potatoes}]_{NP}$.

The power of combinatory rules also allows us to yield *argument cluster coordination*, also known as *non-constituent coordination*, as in Example 2.35. argument cluster non-constituent coordination

(2.35) I made pancakes for Egbert and a salad for Hortense.

The term *non-constituent coordination* stems from the fact that *Egbert cake* and *Hortense a salad*, for instance, are not constituents in a traditional grammar, but rather a *cluster* of the two direct objects of *made*. Steedman [2000] prefers the term *argument cluster coordination*, arguing that it is not possible in general to conjoin non-constituents. A typical CCG analysis for argument cluster coordination is shown in Figure 2.9. For conciseness, assume the following abbreviations: $V \equiv (S\backslash NP)/NP$, $V^2 \equiv ((S\backslash NP)/NP)/NP$, $I \equiv S\backslash NP$.

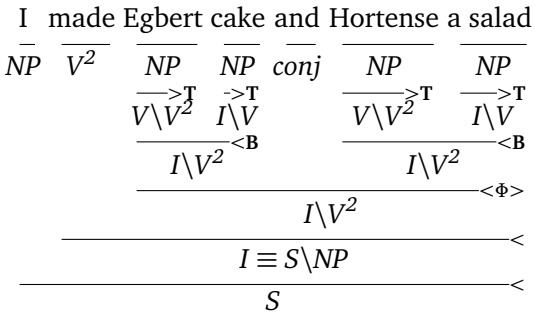


Figure 2.9: Argument cluster coordination

In Steedman's treatment of CCG, coordination is handled by the extra-combinatory rule schema in Rule 2.36, known as *syncategorematic coordination* (Φ). It corresponds directly to our syntactic characterisation of coordination: two like categories with an interposed conjunction forming a single constituent of the same category. syncategorematic coordination

$$X\ conj\ X \Rightarrow X \tag{2.36}$$

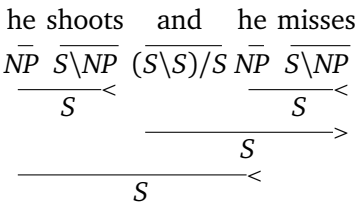


Figure 2.10: Coordination with $(T\backslash T)/T$

Despite its simplicity, this rule schema hides additional complexity. In particular, it matches three categories on the left side, unlike any other combinatory rule. In a parser, the direct implementation of this rule would necessitate a special case, particularly because the CKY-based parsing algorithm commonly used for CCG parsing accommodates only unary and binary production rules.

In an attempt to stay within CCG, we may consider assigning conjunctions the category $(T\backslash T)/T$ for any category T. Such a solution would correctly accept a legitimate case of coordination, as in Figure 2.10. However, consider Example 2.38 below, in which *cooks* combines by backward harmonic composition with the incompletely conjoined *and he cleans*, leading to the acceptance of the ungrammatical sentence[4].

(2.37) [[He cooks]$_S$ and [he cleans]$_S$]$_S$.

(2.38) *He bought a robot that$_{(N\backslash N)/(S\backslash NP)}$ [cooks$_{S\backslash NP}$ [and he cleans]$_{S\backslash S}$]$_{S\backslash NP}$

We now see the justification for the unique form of syncategorematic coordination: such a rule consumes all three categories in a single step, and thus prevents T from combining with surrounding categories. Baldridge [2002] notes that Steedman's syncategorematic coordination is tantamount to implicit use of the application-only mode, limited to the arguments of coordination only. C&C and CCGbank currently use a different strategy for representing, and handling coordination:

$$conj \qquad X \Rightarrow X[conj]$$

$$X \quad X[conj] \Rightarrow X$$

These rules simulate syncategorematic coordination by giving a conjunction which has absorbed one argument to the right (the equivalent of $X\backslash X$ in the pure CCG scheme) the feature $[conj]$, which prevents it from participating in combinatory rules which would otherwise be considered. A category $X[conj]$ then absorbs the category X to the left, yielding the plain category X. While permitting the correct analysis of coordination,

[4]Baldridge [2002] observes that the category $(T\backslash T)/T$ is indeed sufficient to specify coordination in the AB calculus (as introduced in 2.1) without overgeneration, since the only rules available to it are forward and backward application.

these rules prevent CCG from achieving full lexicality, since they lie outside of the kernel of CCG combinatory rules and cannot otherwise be specified through the lexicon. From a parser implementation viewpoint, embedding a particular model of coordination behaviour limits a parser's generality and extensibility to arbitrary languages.

MMCCG, which allows us to restrict the set of combinatory rules in which a given slash may participate, offers a simple solution to the above problem. We give conjunctions a MMCCG category of the form $(T\backslash_\star T)/_\star T$, preventing either slash from participating in any operation but application. Figure 2.11 demonstrates that this assignment blocks the ungrammatical Example 2.38.

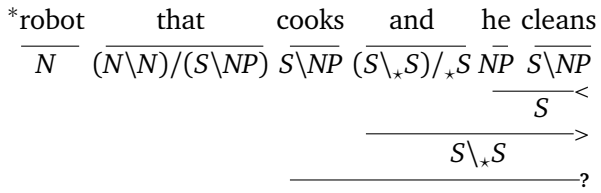


Figure 2.11: Modality blocks overgeneration

Moving the specification of coordination into the lexicon highlights a major benefit of lexicalisation. Consider the case of Japanese, in which two full sentences may be conjoined with the particle *shi*, while nouns are conjoined with the particle *to*. While supporting this behaviour in pure CCG would have required exceptions to be encoded outside the lexicon, we can simply give the below MMCCG categories in the lexicon to define that nouns are to be conjoined with one conjunction, sentences with another.

$$to \vdash (NP\backslash_\star NP)/_\star NP$$
$$shi \vdash (S\backslash_\star S)/_\star S$$

We have demonstrated that MMCCG allows us to remove an extra-combinatory rule from the implementation of a parser, moving the specification of the behaviour into the lexicon where it is encoded as modes. The advantages of increased lexicality are numerous: we can make distinctions in coordination behaviour on the scale of lexical items, and build more efficient parsers which instantiate fewer combinatory rules due to the restrictive power of modes. With MMCCG, we can write entirely generic parser components, free from the extra-combinatory rules and extra-lexical constraints on which pure CCG parsers rely.

CHAPTER 3

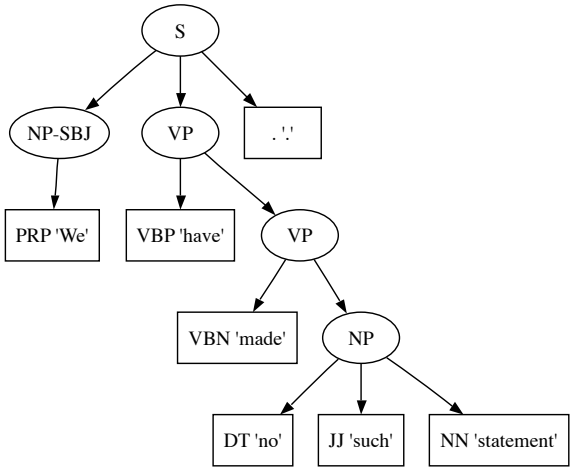
Processing CCGbank

3.1 Introduction

In this chapter, we introduce the key resources and tools of our work, and at the same time describe and evaluate our solution to a fundamental issue with the corpus on which our later work is based. While characterising our approach to this problem, we discuss the structure, conventions and standard evaluation associated with this corpus.

Penn Treebank [Marcus et al., 1994, Taylor et al., 2003] is a landmark in corpus linguistics, as one of the largest annotated corpus collections in the English language. Containing a corpus of 3 million words annotated for generic phrase structure grammar, it is a key resource for statistical parsing.

A typical Penn Treebank derivation, and its induced tree structure is shown in Figure 3.1. Penn Treebank derivations typically induce shallow trees, with phrase-internal structure (for example, *no such statement* in Figure 3.1) minimal or non-existent. Furthermore, the set of Penn Treebank POS tags encode additional syntactic and semantic information, such as grammatical relations (for example, the tag `NP-SBJ`). Finally, the gaps left by constituents moved by syntactic processes are marked by null elements known as *traces*.




```
( (S
    (NP-SBJ (PRP We) )
    (VP (VBP have)
     (VP (VBN made)
      (NP (DT no)
          (JJ such)
          (NN statement) )))
    (. .) ))
```

Figure 3.1: Penn Treebank derivations

Penn Treebank has enabled the construction of corpora built for other grammar formalisms, including HPSG [Miyao et al., 2004] and LTAG [Chen et al., 2006], often serving as the source dataset of an automatic corpus induction process. The development of automatic or semi-automatic methods for deriving new corpora from Penn Treebank is an attractive means to the end of wide-coverage parsing in a given formalism, due to its size and coverage, while obviating the enormous cost of manually re-annotating a corpus for different formalisms.

CCGbank, a 1.2 million word CCG corpus drawn from the Wall Street Journal section of Penn Treebank, is the training and evaluation corpus for the C&C parser, as well as an example of a corpus derived from Penn Treebank through an automatic procedure [Hockenmaier and Steedman, 2005]. CCGbank is the only corpus of its size and coverage for CCG, and its availability is the impetus for a great deal of the interest in CCG as a grammar formalism for practical, efficient parsing.

Not every derivation in Penn Treebank is represented in CCGbank; Hockenmaier reports a coverage of 48,934 of 49,208 derivations (99.44%). The remaining derivations are missing either because they exhibit a gapping construction which cannot be analysed at all by the combinatory rules specified by CCG, or otherwise because the corpus conversion process yielded an invalid CCG derivation.

3.2 CCGbank representations

The structure of CCGbank mirrors that of its source corpus. It is divided into 25 *sections*, each of which contains a series of *documents*. In turn, each document contains a series of *derivations*, which are typically drawn from the same source article. For each derivation, CCGbank provides its bracketed parse and its dependency data. The dependency-based evaluation methodology employed in this work, following that of Clark and Hockenmaier [2002], uses this dependency data as a gold standard. Figure 3.2 depicts the two items comprising each CCGbank derivation.

We describe the format of the dependency information, which specifies the word-word dependencies that hold in a derivation. A dependency relation is a 6-tuple, consisting of:

- i_1, l_1: the index, and lexical item of the argument
- i_2, l_2: the index, and lexical item of the head
- c: the head category
- r: the index of the argument in category c satisfied by this relation

For example, the second row in Figure 3.2 indicates that the lexical item *We* satisfies argument 1 ($S[pt]\backslash NP_1)/NP_2$ of the category of *have*. The parser evaluation methodology of Clark and Hockenmaier [2002], which relies on the computation of a F-score

```
(<T S[dcl] 0 2>
  (<T S[dcl] 1 2>
    (<L NP PRP PRP We NP>)
    (<T S[dcl]\NP 0 2>
      (<L (S[dcl]\NP)/(S[pt]\NP) VBP VBP have ...>)
      (<T S[pt]\NP 0 2>
        (<L (S[pt]\NP)/NP VBN VBN made ...>)
        (<T NP 1 2>
          (<L NP[nb]/N DT DT no ...>)
          (<T N 1 2>
            (<L N/N JJ JJ such ...>)
            (<L N NN NN statement N>))))))
  (<L . . . . .>))
```

i_1	i_2	c	r	l_1	l_2
0	1	$(S[dcl]\backslash NP)/(S[pt]\backslash NP)$	1	We	have
0	2	$(S[pt]\backslash NP)/NP$	1	We	made <XB>
2	1	$(S[dcl]\backslash NP)/(S[pt]\backslash NP)$	2	made	have
5	2	$(S[pt]\backslash NP)/NP$	2	statement	made
5	3	$NP[nb]/N$	1	statement	no
5	4	N/N	1	statement	such

Figure 3.2: A CCGbank derivation and its dependency information

based on the gold standard equivalence of these dependency tuples, elevates semantic correctness over structural identity.

Where G is the set of gold standard tuples, and C is the set of candidate tuples, we calculate P, R and F – precision, recall and F-score, as follows:

$$P = \frac{|C \cap G|}{|C|} \tag{3.1}$$

$$R = \frac{|C \cap G|}{|G|} \tag{3.2}$$

$$F = \frac{2PR}{P+R} \tag{3.3}$$

These are standard evaluation metrics common to many NLP and information retrieval tasks [Manning and Schütze, 1999]. Precision is the proportion of dependencies present in both the candidate and gold standard, to the total number of dependencies in the candidate parse, a measure of how well we have captured *only* the dependencies which are relevant. Recall is the proportion of common dependencies to the total number of gold standard dependencies, a measure of how many of the gold standard dependencies we have successfully captured. The F-score is the harmonic mean of precision and recall,

and is a common summary statistic.

If the evaluation is *unlabelled*, then the intersection is computed against a reduced tuple of the head and argument lexical items only $\langle l_1, l_2 \rangle$, and not the complete tuple $\langle c, r, l_1, l_2 \rangle$.

Instead of the dependency-based approach of the above evaluation, PARSEVAL, an evaluation metric which has dominated in parser research, punishes the structural property of *crossing brackets*: if we imagine that the grouping induced by a constituent in the candidate parse is enclosed by brackets, PARSEVAL punishes a candidate grouping which overlaps some grouping in the gold standard. Several flaws in this structurally motivated evaluation metric have been pointed out [Carroll et al., 1999]. It is argued that PARSEVAL fails to capture the correctness of a parse in shallow treebanks such as the Penn Treebank, since such treebanks are characterised by fewer brackets, and hence fewer opportunities for crossing brackets. The emphasis on structural identity is particularly problematic for CCG, due to the spurious ambiguity property, as introduced in Section 2.2.4.

Furthermore, its dominance as the *de facto* metric requires parsers which produce more detailed forms of dependency data to discard much of that information in a PARSEVAL frontend to their evaluation framework. These disadvantages have motivated the adoption of a *dependency-based evaluation* as a common means of evaluating CCG parser performance, which we present in the following section.

3.2.1 C&C parser evaluation

L	P R F	Labelled precision, recall and F-score
LF*		Labelled F-score with C&C assigning POS tags
LSA		Labelled sentence accuracy
U	P R F	Unlabelled precision, recall and F-score
SUP		Supertagger accuracy
COV		Coverage

Table 3.1: C&C evaluation metrics

The dependency-based evaluation used by C&C considers the set of dependencies induced by the candidate parse, comparing them to those of a gold standard. Table 3.1 shows the values computed during the evaluation of the C&C parser. Precision, recall and F-score are calculated in two ways: *unlabelled*, which deems a candidate dependency to match a gold standard dependency if their lexical items match (l_1 and l_2 in Figure 3.2),

and *labelled*, which also takes into account the head category c, and the argument slot r satisfied by the argument l_1.

We also consider the parser's labelled F-score when POS tags are assigned by C&C, instead of taken from the gold standard (LF*), as well as the proportion of derivations for which 100% of the dependencies have been correctly captured (LSA). Finally, we consider the proportion of categories correctly assigned to tokens by the supertagger (SUP), and the proportion of derivations for which C&C succeeds in finding a spanning analysis (COV).

3.3 CCGbank preprocessing and associated problems

Hockenmaier describes a sequence of heuristics employed to correct errors inherent in the source corpus which adversely affect the category assignment process. One of the preprocessing steps is to strip quote symbols (‘‘, ’’, ‘, ’) from the corpus. The CCGbank conversion process strips the corpus of quotes to simplify the corpus conversion algorithm: their presence would require numerous special cases throughout the process, including the head-identification algorithm, motivating their removal in the generated CCGbank [Hockenmaier, personal communication]. Although it was intended that quotes would be re-instated in post-processing, this was never achieved.

From the vantage point of practical parsing, the absence of quotes is not satisfactory, since a parser trained on CCGbank cannot acquire a statistical model of the distribution of quote symbols in English.

The frequency of Penn Treebank derivations containing *candidate quotes* (quote symbols which are candidates for quote re-insertion) in each of the sub-corpora of the usual CCGbank corpus split, is given in Table 3.2. Although CCGbank is drawn from the genre of newswire, and hence is somewhat skewed towards heavy use of quotes, the impact of failing to process quotes on the coverage of a parser is considerable, given their wide distribution through general English text.

Section	*Frequency of quotes*	
Development (00)	361/1921	(18.79%)
Training (02-21)	6702/41753	(16.05%)
All (00-24)	7941/49208	(16.14%)

Table 3.2: Frequency of candidate quotes in Sections 00, 02-21 and all of CCGbank

The C&C parser, to be introduced in Section 2.3 as the target of our implementation of multi-modal CCG, employs the stopgap measure of introducing *absorption rules*, which

absorb the quote with no effect on the resulting category. That is, for any category X:

$$“\ X \Rightarrow X \tag{3.4}$$

$$X\ ” \Rightarrow X \tag{3.5}$$

Although this measure is sufficient to ensure that the parser does not completely fail to analyse input containing quotes, the absence of quotes in the training data means that the supertagger and parser are unable to make use of contextual cues provided by the presence of quotes. The supertagger component of C&C solves a tagging problem, where the tokens are lexical items, and the tags are sets of categories. Intuitively, a supertagger assigns each input token a set of categories which it is likely to take. Supertaggers enable an increase in efficiency, since although a typical lexical item may be mapped to a number of categories, the information provided by local context as well as past history may rule out a number of these alternatives. The same insight underlies Hidden Markov Model POS and named entity tagging. The C&C supertagger incorporates a conditional maximum entropy model, which models the context of a given token as a feature vector of contextual predicates [Clark, 2002]. In this case, a contextual predicate is any feature which may yield information as to the correct assignment of a category. The presence of quotes is a contextual predicate which is currently unavailable to the supertagger.

Consider several categories attested in CCGbank for the lexical item *said*, listed in Figure 3.3.

said	$\vdash$	$(S[dcl]\backslash NP)/NP$	*Egbert said a prayer.*
	$\vdash$	$(S[dcl]\backslash NP)/S[qem]$	*The teacher said why the sun shines.*
	$\vdash$	$((S[dcl]\backslash NP)/S[dcl])/PP$	*Hortense said to the teacher, “Egbert ate the cake.”*
	$\vdash$	$(S[dcl]\backslash NP)/S[dcl]$	*Hortense said, “Egbert stole the cookie.”*
	$\vdash$	$(S[dcl]\backslash NP)/S[q]$	*Hortense said, “Why does the sun shine?”*

Figure 3.3: Selected categories for *said* in CCGbank

A supertagger trained on a corpus which does not incorporate quote symbols is unable to take advantage of their presence. A supertagger could otherwise utilise the local information given by quotes by preferring to assign one of the last three categories, depending on the values of other contextual predicates, to anticipate that a full sentence or question (typically enclosed in quotes) will follow the verb. This additional information allows the supertagger to make more efficient use of resources by reducing the number of categories the parser must subsequently consider.

In this section, we describe an algorithm for the reintroduction of quotes to CCGbank, evaluating the impact of our changes against the C&C parser which we will modify for multi-modal CCG in the latter part of this work.

3.4 Reintroducing quotes

The general structure of our approach is as follows: for each pair of a Penn Treebank derivation and its corresponding CCGbank derivation, locate nodes in the Penn Treebank derivation corresponding to the quote symbols. Then, for an opening quote, we identify the first leaf which would follow the opening quote, and for a closing quote, the first leaf which would precede it. The nodes corresponding to this pair span the quoted text. We then use an algorithm which determines where in the tree the quote leaf should be inserted. Finally, we must finally increment the indices in the dependency information (i_1, i_2 in Figure 3.2) as appropriate to reflect the addition of the new quote token.

A general framework for our re-quoting mechanism is given in Figure 3.4. In the algorithm, P and C represent the two corpora Penn Treebank and CCGbank, while c and d are a CCGbank derivation and its dependency information.

```
1: for (p, (c, d)) ∈ MAP-CORPORA(P, C) do
2:     C ← FIX-TREE(C)
3:     I ← FIND-QUOTE-INDICES(P)
4:     for each (begin, end) ∈ I do
5:         b, e ← GET-CCG-NODES(C, begin, end)
6:         n ← FIND-ATTACH-NODE(C, begin, end)
7:         ATTACH-QUOTE-STRUCTURE-AT(n)
8:         d ← FIX-DEPENDENCY-INFO(d)
9:     end for
10: end for
```

Figure 3.4: General framework for reinstating quotes

3.4.1 Alignment

Firstly, we perform an *alignment* between the two corpora. As CCGbank is a subset of the WSJ section of Penn Treebank, this entails the removal of those Penn Treebank derivations which do not correspond to any CCGbank derivation. We match up derivations according to the text in their leaves; two derivations are said to correspond if the text of the Penn Treebank derivation, stripped of elements which are not retained by the CCGbank conversion process, is identical to that of the CCGbank derivation. The following Penn Treebank tokens are removed by the CCGbank conversion process:

- Any leaf beginning with *: these are traces
- ‘, ’, ‘‘, ’’, where the POS tag is not POS

A trace marks the node from which a constituent is moved by some syntactic construction, and does not correspond to a token in the original text. A trace may carry an index which co-refers to the slot at which the constituent is now located. Since trace

information is integrated into the feature structure of a CCG category by the CCGbank conversion process, no corresponding node will be found in the induced CCG derivation.

Finally, since our goal is to reinstate the quotes removed by the conversion, such tokens will not be found in the original CCGbank corpus. The only subtlety lies in the sense of ’ as an allomorph[1] of the English possessive morpheme *’s*; we disambiguate this from a single close quote by its POS tag, which is POS for the possessive case and ’ for a close quote.

3.4.2 Tree transformations

Next, we address an artifact of the CCGbank conversion process which would otherwise affect our analysis. There is some variation between American and British English as to whether punctuation following a span of quoted text, such as a comma, should lie between or after the quotes.

(3.6) ‘Oh, Jeeves,’ I said. ‘Cats! What about it? Are there any cats in the flat?’

(3.7) There are eleven songs, including “Black.”

(3.8) There will be a matinee of “Who’s Afraid Of Virginia Woolf?”.

In American English, this punctuation lies within the quoted span at all times, as in Example 3.7 [University of Chicago Press, 2001]. However, in British English, this only applies when the punctuation is part of the quote itself, as in Example 3.8. In both conventions, the comma separating the quoted from unquoted portion is also considered to constitute part of the quote.

> Periods and commas precede closing quotation marks, whether double or single. This is a traditional style, in use well before the first edition of this manual (1906). As nicely expressed in William Strunk Jr. and E. B. White’s Elements of Style, “Typographical usage dictates that the comma be inside the [quotation] marks, though logically it often seems not to belong there” (p. 36; see bibliog. 1.1). The same goes for the period.
>
> (The Chicago Manual of Style, 6.5: Periods and commas.)

In a CCGbank analysis, the node containing the absorbed punctuation token is typically the leftmost leaf of a right subtree, as shown in Figure 3.5, while the quoted text is contained in the left subtree. Figure 3.6a demonstrates the default analysis of the quote comma, while Figure 3.6b shows the undesirable effect when a quoting procedure is applied to this default analysis. α represents the subtree containing the quoted span of text. As shown, the close quote is stranded in the right subtree, separated from the body of the quoted span.

[1] A single morpheme can have multiple phonological realisations depending on its context, each of which is known as an *allomorph*. For example, the allomorphs of the English plural morpheme *-s* not only include *-s* (e.g. *cake – cakes*), but also the empty allomorph ∅ (e.g. *sheep – sheep*).

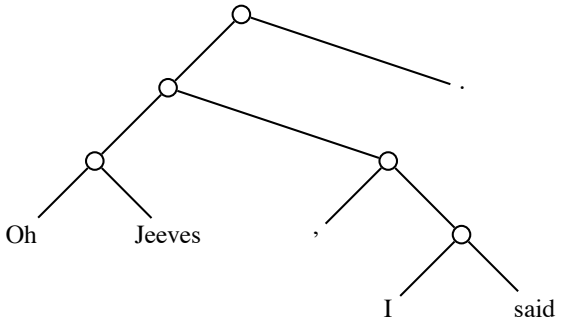
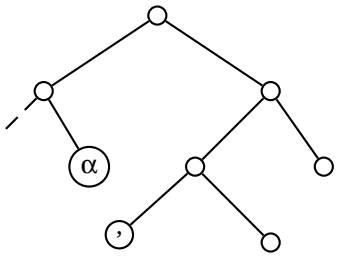
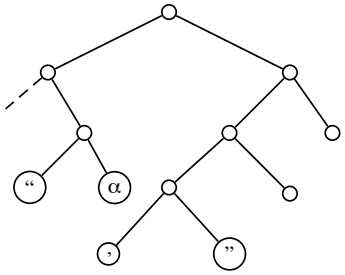


Figure 3.5: Analysis of quote commas in CCGbank

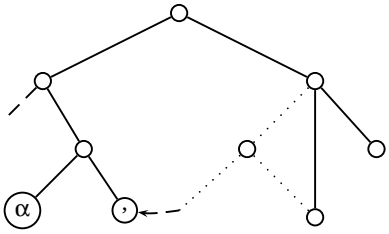
Ideally, this comma should be situated on the left subtree, such that a quoting structure can correctly enclose the quoted subtree, including this absorbed comma. The transformation, effected by our procedure FIX-TREE, is depicted in Figure 3.6c, and the improved analysis which FIX-TREE allows is shown in Figure 3.6d.



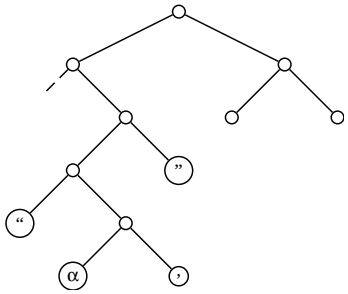
(a) Before SHIFT



(b) Close quote stranded in right subtree



(c) After SHIFT



(d) Correct analysis after SHIFT

Figure 3.6: Effect of the tree transformation SHIFT

3.4.3 Reattachment of quotes

For each CCGbank derivation and its corresponding Penn Treebank derivation, we obtain the indices delimiting a span of quoted text in the corresponding Penn Treebank derivation. For each pair of span indices, we locate the corresponding leaves in the CCGbank derivation, and determine the depth in the tree at which the quotes are to be re-attached (FIND-ATTACH-NODE). Finally, we update the indices in the dependency data for the derivation.

Two properties are desirable in a specific implementation of our approach. Firstly, inserting the quotes should yield exactly the text of the original derivation. Furthermore, the node at which we attach the quoting structure should be the highest node which spans exactly the quoted portion of text, for maximum generality.

We will evaluate each of the FIND-ATTACH-NODE implementations in terms of how well they fulfill these two properties. For example, an implementation which simply attaches the quote nodes immediately above the leaves certainly satisfies the first property, but fails to satisfy the second.

We can compute the number of derivations for which the algorithm inserts the quotes while preserving the sequence of tokens of the original text, to determine how well we adhere to the first criterion. We will also examine the algorithms to establish how well they satisfy the second criterion.

Algorithm LCA

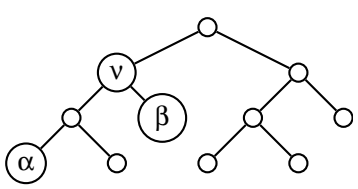
LCA chooses as the attachment node the least common ancestor of the two nodes that span the quoted portion. The least common ancestor of two nodes is defined as the ancestor of both nodes which is highest in the tree [Cormen et al., 2001]. We considered this as a first approach to achieving the second criterion of generalisation.

Algorithm 1 (LCA): FIND-ATTACH-NODE($begin, end$)

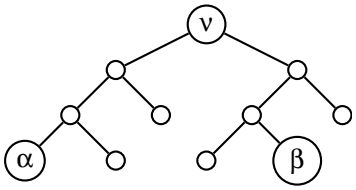
1: **return** LCA($begin, end$)

However, this strategy does not satisfy the first criterion of maintaining the sequence of tokens from the original text, in particular when the beginning and ending nodes are not respectively the leftmost and rightmost leaves of some subtree.

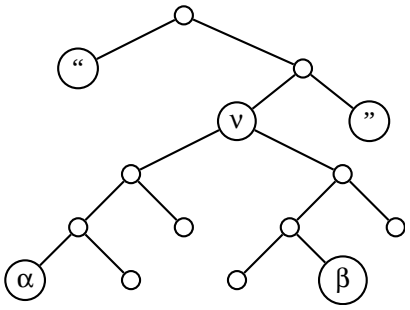
Figure 3.7b demonstrates the resulting analysis when this property is not satisfied: in this example, the entire subtree under node ν (the LCA of nodes α and β), will be incorrectly enclosed in quotes.



(a) LCA succeeds when α and β delimit a subtree



(b) LCA fails when this does not hold



(c) Incorrectly quoted derivation after LCA analysis

Figure 3.7: LCA analysis

Instances of such structures, as seen in Example 3.9, are particularly common in newswire, where the convention is to only enclose in quote symbols a *verbatim* quote, allowing readers to distinguish a paraphrase of an exact quote from its original.

(3.9) Polls once named Tokyo Giants star Tatsunori Hara, a “humble, uncomplaining, obedient soul,” as the male symbol of Japan.

When the quoted text lies exactly within some subtree, then the least common ancestor of the two nodes delimiting the text is indeed the highest node in the tree which contains strictly the quoted text. However, this approach is not robust, since the least common ancestor when this condition is not met may contain tokens that were not in the original span of quoted text, as in Figure 3.7c.

Of the 8677 derivations in CCGbank containing quotes, LCA correctly re-inserts the quotes while preserving the correct order of the Penn Treebank text in only 4929 (56.8%) of cases. The cases upon which LCA fails include derivations similar to Figure 3.7b, in addition to derivations with unmatched quotes. These inadequacies in LCA motivate our consideration of an alternative algorithm.

Algorithm SPAN

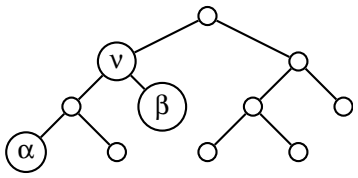
Since there are spans of quoted text which do not lie exactly under some subtree, it is not always possible to attach quotes that sandwich the text. SPAN addresses this by allowing the opening and closing quotes to (potentially) attach above different nodes, by considering them separately.

Let us consider opening quote insertion. Initially, set the candidate node cur to the node $begin$ itself. When the loop terminates, cur will be the node above which to insert the quote. Then, as long as the text under cur's parent is some contiguous subsequence of the span of text delimited by $begin$ and end, ascend to the parent. At the beginning of each iteration, the text under cur is some contiguous span of the quoted text. When the loop terminates, cur is the highest node which contains only a contiguous subsequence of the quoted text. Performing the analogous procedure for the closing quote, we obtain appropriate nodes above which to insert each of the opening and closing quotes.

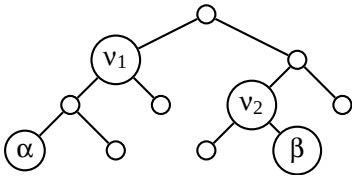
Algorithm 2 (SPAN): FIND-ATTACH-NODE($begin, end$)

```
1: attachment-nodes ← ⟨⟩
2: for each node ∈ {begin, end} do
3:     cur ← node
4:     while cur.parent ≠ nil and MATCH?(TEXT(cur.parent), quoted) do
5:         cur ← cur.parent
6:     end while
7:     APPEND(cur, attachment-nodes)
8: end for
9: return attachment-nodes
```

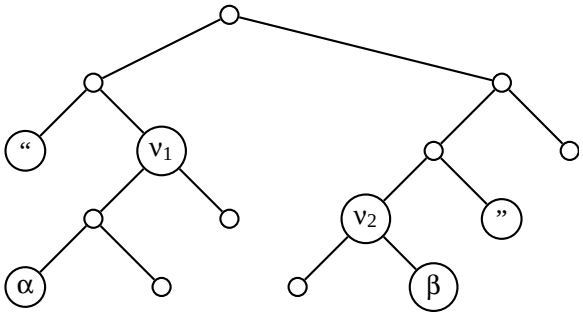
SPAN correctly reinstates 8477 of the 8677 occurrences (97.7%) of candidate quotes. The remaining cases either involve single quotes nested within a span of double quotes, a limitation of our particular implementation, or are otherwise noise cases, in which a derivation begins with a closing quote, or ends with an opening quote.



(a) SPAN analysis is identical to LCA when the same-subtree condition holds (v_1 and v_2 coincide)



(b) SPAN still yields a meaningful analysis when the condition is violated



(c) Quoted derivation after SPAN analysis

Figure 3.8: SPAN analysis

We ultimately choose SPAN over LCA, as it satisfies the two criteria of generalisation and preservation of the sequence of tokens, and is robust in the face of a quoted span which is not confined to the leaves of a subtree.

3.4.4 Other approaches considered

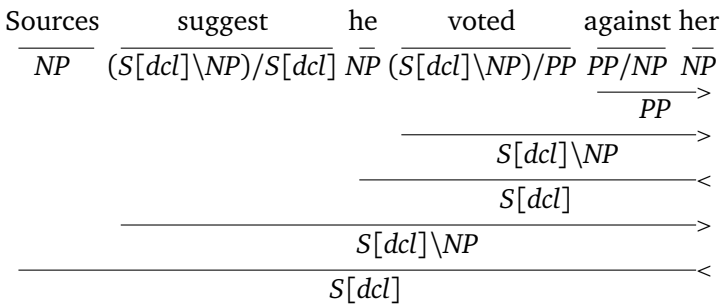
In a SPAN analysis where the nodes delimiting the quoted span are not the leftmost and rightmost leaves of the same subtree, we consider inserting the opening and closing quotes separately. Is it possible to perform a tree transformation similar to SHIFT (shown in Figure 3.6) so that the quoted span once again lies completely within the same subtree?

Consider the following examples, which exemplify the newswire convention mentioned in 3.4.3 of enclosing in quotes only the span of text taken *verbatim* from its source.

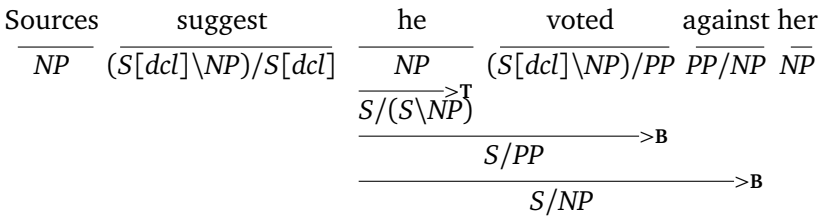
(3.10) Sources suggest he voted against her.

(3.11) Sources suggest “he voted against” her.

It is clear that if the quoted span lies completely within the same subtree, we must be able to derive a category which covers exactly the quoted span. By employing spurious ambiguity (see Section 2.2.4), we can do precisely this:



(a) Normal form analysis (Example 3.10)



(b) Non-normal form analysis (Example 3.11, *partial*)

Figure 3.9: Preserving the same-subtree property through spurious ambiguity

In Example 3.11, we derive the category $S[dcl]/NP$ for the span of text *he voted against*, which would yield the derivation tree in Figure 3.10, permitting us to enclose exactly the subtree containing those words in quotes.

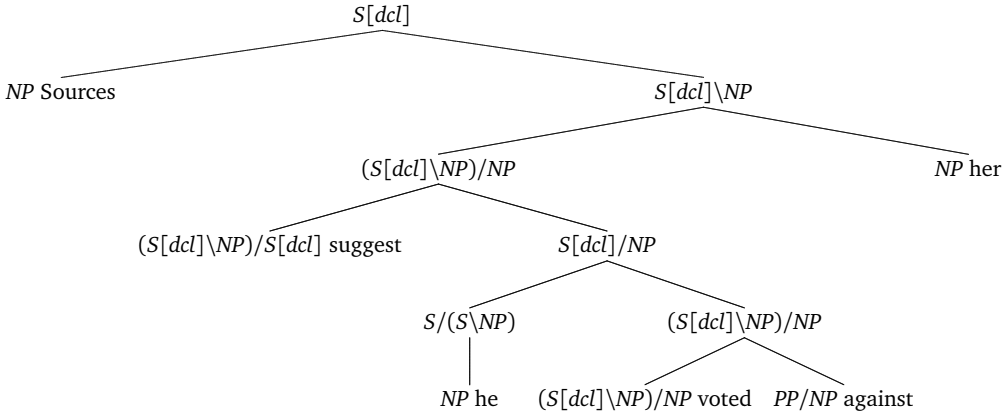


Figure 3.10: Full derivation tree for Example 3.11

This derivation through spurious ambiguity allows us to exactly enclose the subtree containing the quoted text *he voted against*. However, this is an unsatisfactory approach for two reasons. Firstly, the presence of non-canonical derivations such as

Figure 3.10 adds additional ambiguity to the corpus: consider that the derivation structures underlying Examples 3.10 and 3.11 are now very distinct. This ambiguity is undesirable in a corpus used to train the statistical model of a parser. Secondly, such a corpus would no longer satisfy the Eisner normal form constraint, which as discussed in Section 2.2.4, grants a CCG parser a considerable increase in efficiency by reducing the number of spuriously ambiguous parses it must consider.

Clearly, no approach which relies on manipulating the deep structure of the derivations to this degree is desirable. For these reasons, we settle for the compromise offered by Algorithm SPAN between preserving the normal-form property of the derivations, and the inability to entirely enclose the originally quoted span.

3.4.5 Preprocessing

We also consider the impact of disabling FIX-TREE, leaving the comma leaf in its place in the right subtree. In Example 3.5, this would result in the text *Oh Jeeves* being quoted in the output. Since the functional effect of this implementation is to swap the order of the comma and the close quote in the re-quoted output, we label this corpus version SWAP, with the tree-transforming version known as SHIFT. While SHIFT preserves the original quoted order of the text, SWAP swaps the position of the inserted close quote and the comma.

(3.12) "The curse is come upon me," cried the Lady of Shalott.

(3.13) "The curse is come upon me", cried the Lady of Shalott.

The impact of each of these two approaches on parser evaluation is quantified in Section 3.4.6.

3.4.6 Experiments and results

We have presented two strategies for selecting the node above which a quoting structure should be attached, LCA and SPAN. LCA fails to yield a correct analysis when no single node spans exactly the quoted text, motivating an alternative approach, SPAN, in which the attachment nodes for the open quote and close quote can differ.

We measure parser accuracy by computing the F-score over these tuples relative to a gold standard, and employ a standard split of the CCGbank, training on Sections 02-21, and evaluating on Section 00. The parser metrics shown in the results table are described in Section 3.2.1.

We note that all of the re-quoted corpora yield an increase in supertagger accuracy, as per our conjecture, while exhibiting a very small decrease in F-score. We have the following hypothesis for the observed results. The supertagging problem deals with assigning categories to a linear sequence of tokens, using contextual predicates which capture features of the neighbourhood of the lexical item for which we want to find a category. The C&C supertagger uses as a feature the tokens in a window of size 2 on either

Corpus	LP	LR	LF	LF*	LSA	UP	UR	UF	SUP	COV
C&C orig	85.53	84.71	85.12	83.38	32.14	92.37	91.49	91.93	93.05	99.06
SHIFT LCA	85.27	84.42	84.85	83.18	32.15	92.15	91.23	91.69	93.17	99.01
SHIFT SPAN	85.30	84.50	84.90	83.38	32.77	92.16	91.30	91.73	93.13	99.06
SWAP LCA	85.19	84.40	84.79	83.22	32.00	92.06	91.21	91.63	93.13	99.01
SWAP SPAN	85.44	84.63	85.03	83.45	32.81	92.22	91.34	91.78	93.18	98.95

Table 3.3: C&C parser evaluation on modified corpora

side of the candidate token, so that the re-added quotes are available to the supertagger as extra information.

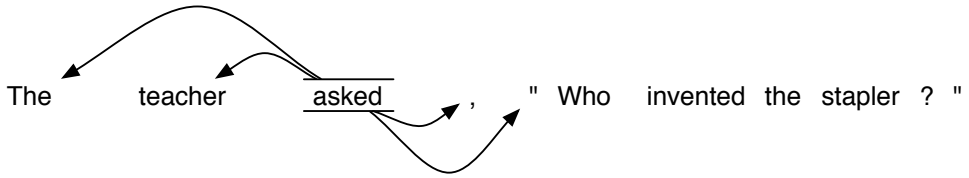


Figure 3.11: Supertagging with quotes as contextual predicates

On the other hand, the extra information provided by quotes is to some extent offset by the fact that the interposed quotes pushes out tokens which would otherwise lie within the supertagger window of two tokens to either side.

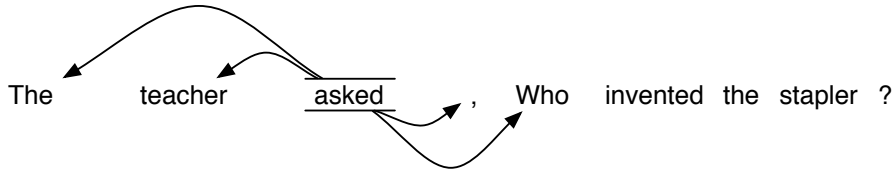


Figure 3.12: Impact of inserted quotes on history window

In a situation such as that depicted in Figure 3.12, the addition of quotes means that the valuable information provided by the association of a *wh*-word (such as *who*, *what*, *when*, *how*) becomes inaccessible as contextual information, which is a key problem. Although the extent of our exploration in the present work is limited to the parser component of C&C, for future work, we suggest performing an evaluation of the impact of the size of the context window on a parser trained on the re-quoted corpus. An significant increase in parser accuracy relative to the evaluation on the baseline window size of two would provide evidence that the supertagger should be modified in response to our corpus transformation.

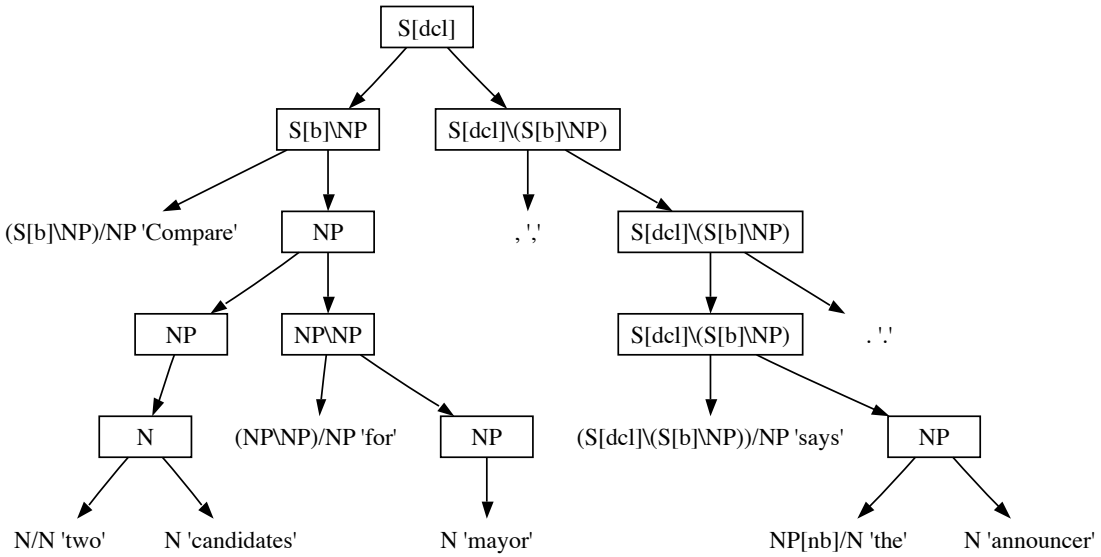
To explain the above results, we conjecture that the extra information provided by quotes to some extent counteracts the information from the tokens which no longer lie within the context window. The net effect on the corpus is, as shown in Table 3.3, a slight increase in supertagger accuracy.

The F-score from the dependency-based analysis is slightly decreased relative to the evaluation on the unmodified CCGbank. The intuition behind this fact is simply that increasing the length of the input increases the load on the parser [Church and Patil, 1982], so that the presence of quotes increases the number of possible attachments which the parser must consider.

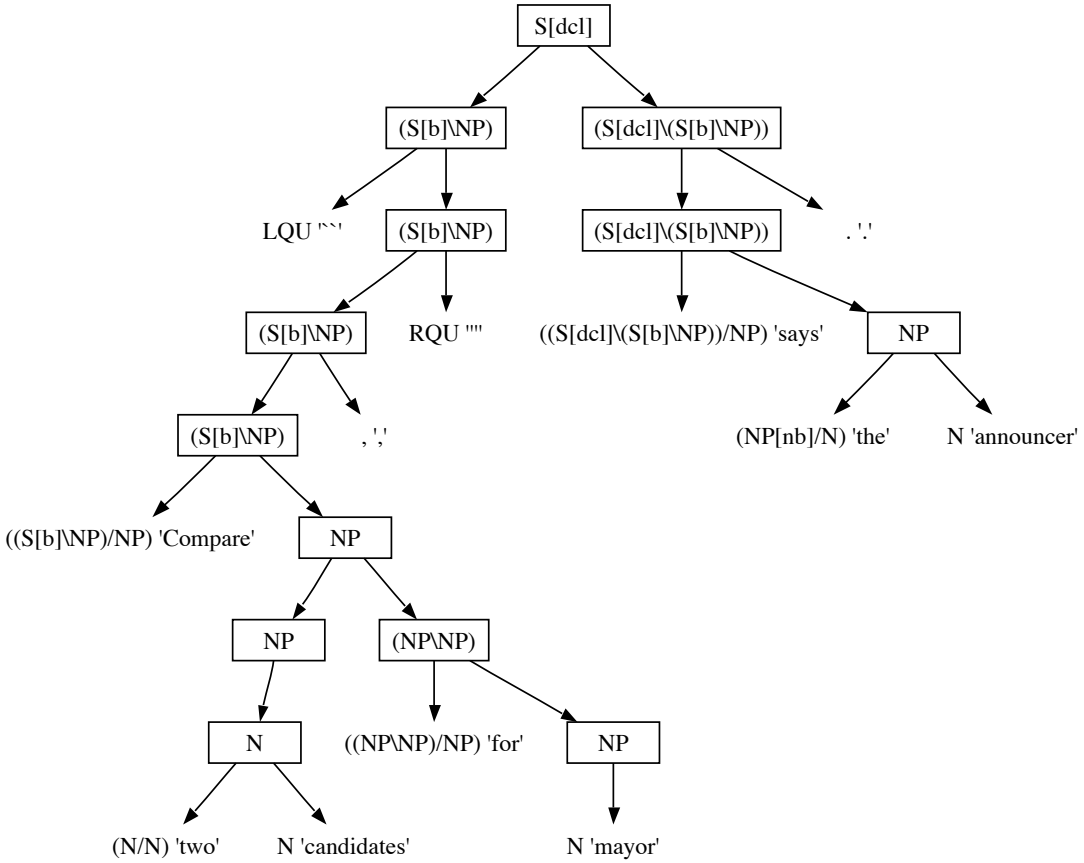
3.5 Conclusion

We have developed a complete and practical algorithm for recovering useful information in the form of quotes, and returning them to a wide-coverage corpus. We have evaluated this re-quoted corpus on a C&C parser and examined its impact on the standard parser evaluation. Key corpus applications such as *speaker* or *topic segmentation*, the identification of changes in speaker or writer throughout a document, may benefit from the additional information we have restored to the corpus.

Through the work described in this chapter, we now have a more complete and more useful resource: a corpus which better reflects the true distribution of English text.



(a) The original, unquoted derivation.



(b) A re-quoted derivation. Note the movement of the comma leaf.

Figure 3.13: Reinstatement of quotes

CHAPTER 4

Deriving a multi-modal CCG corpus

The primary goal of this work is to increase the amount of information available in the corpus to the parser, without decreasing the coverage of the original corpus (and hence the parser). We supplement the corpus with restrictions, in the form of *modes*, which decrease the space of combinatory rule instantiations that a parser must consider. Our strategy for adding modality annotations while preserving the correctness of existing CCGbank derivations is to strongly discourage the assignment of a mode which would block a combinatory rule used in some CCGbank analysis.

As a corpus that is automatically derived, as opposed to manually annotated, CCGbank contains many *noisy derivations*: faulty analyses which are artifacts of Hockenmaier's corpus conversion process. An example of a noisy derivation is given in Figure 4.1.

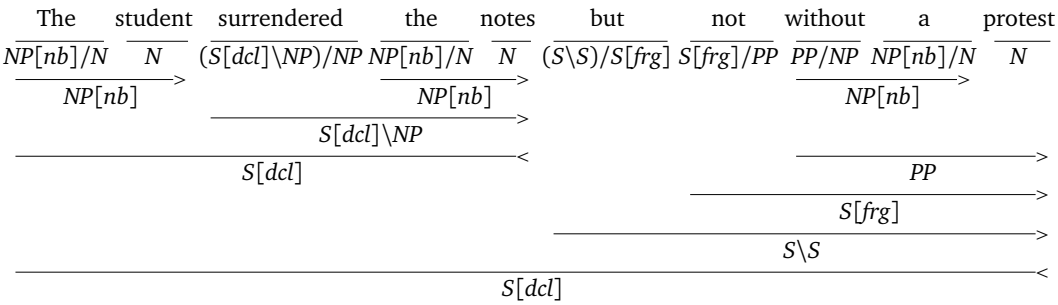


Figure 4.1: A noisy CCGbank derivation (CCGbank 0.44.9)

In this derivation, *not without a protest* is misconstrued as a sentence fragment with category $S[frg]$, causing *but* to also receive the noisy category $(S\backslash S)/S[frg]$ in an attempt to preserve a spanning analysis. Noise from a number of sources – noisy combinatory rule applications, noisy categories, and noisy derivations, together complicate the processing necessary for our own corpus derivation task.

In order to minimise the impact of noise on our corpus analysis, we must rely on the assumption that noisy analyses are rare relative to correct ones. Suppose a particular slash is used with composition 50 times, and is used with application 950 times. As we will show, the fact that composition only accounts for 5% of the cases is not sufficient evidence to exclude the composition cases as noise, nor is it sufficient evidence to

conclude that the composition cases are legitimate. In this case, we should not dismiss these analyses as noise simply because they are uncommon: for example, coordination of the kind shown in Figure 4.2 is rare, but entirely grammatical. On the other hand, a rare analysis may indeed be an instance of noise.

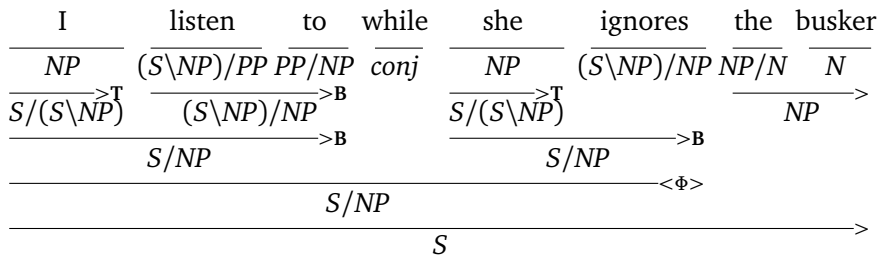


Figure 4.2: Rare but correct analysis

We describe a semi-supervised approach which involves both manual annotation for some categories and the automatic acquisition of modes for others. We will show that such an approach yields a suitable compromise between the two extremes of the expensive manual annotation of the entire category set, and the reliance on algorithmic means with its attendant decrease in precision. Our approach to deriving a multi-modal CCG corpus is iterative, under two constraints: we wish to increase the restrictiveness of the corpus, while minimising the number of regressions (broken derivations) it causes. Finally, we will describe and perform an evaluation on the corpora produced by the work of this section, to determine its effect on the difficulty of the supertagging task.

First, we formulate the mode assignment problem, and describe our own version of Baldridge's formulation of multi-modal CCG which we have developed for this work. We then consider each of the modes in our mode scheme, justifying their effect, explaining their place in our system and comparing them to other possible analyses.

4.1 Formulation of the mode assignment problem

Intuitively, the mode assignment problem consists of annotating each slash of each category of each lexical item in the corpus with a mode. Structurally, MMCCG does not allow a bare slash, so every slash must be assigned some mode. Functionally, however, the maximally permissive mode $(\cdot)$ will license every combinatory rule for a given slash.

In our formulation of MMCCG, we consider the reduced, three-mode system of Figure 4.3. The implementation of a reduced mode hierarchy will nevertheless allow us to survey the structural difficulty of implementing MMCCG within C&C, and devise an annotation strategy with a view to extending it eventually to the full hierarchy of modes as presented by Baldridge [2002].

Figure 4.3: Three-mode system

4.2 Comparison to the MMCCG approach of Baldridge

We have modified the version of the formalism as presented in Baldridge [2002] to satisfy a number of constraints: our own intuition into the structure of CCGbank, the scope of this present work, the structure and implementation details of the C&C parser, and the accessibility of the material.

In this section, we will explain in detail the modifications we have made to Baldridge's presentation of MMCCG, and justify the choices we have made.

4.2.1 Differences in theory

Baldridge posits a system of seven modes, as shown in Figure 4.4. Compared to our simplified scheme as represented in Figure 4.3, Baldridge's scheme provides more granularity in precisely specifying the behaviour of crossed composition. Crossed composition is the recipient of additional derivational control because, as we will see in Section 4.3.4, its distribution in a language is typically highly restricted due to its ability to induce undesired word orders. This gives lexicon writers a fine brush with which to separate grammaticality from ungrammaticality where it is most needed.

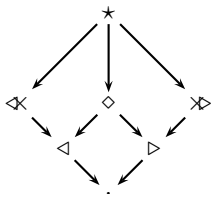


Figure 4.4: Baldridge's hierarchy of modes

Baldridge [2002] shows that the modes which provide a restricted degree of crossed composition $\diamond\!\!\times$ and $\times\!\!\diamond$ are useful in preventing cases of overgeneration resulting from object extraction. In order to explain the overgeneration, we introduce further terminology from linguistics. Recall from Section 2.1 that a ditransitive verb takes one subject argument and two direct object arguments. Common ditransitive verbs include *give*, *buy* and *sell*. Ditransitive verbs are very often verbs of *transfer of ownership*.

(4.1) I sold him a goldfish.

(4.2) Hortense gave Egbert a paper donkey.

Typically, the category of a ditransitive verb is $((S\backslash NP)/NP)/NP$. However, the two direct object arguments are not interchangeable: after all, the below sentences do not have the same semantics.

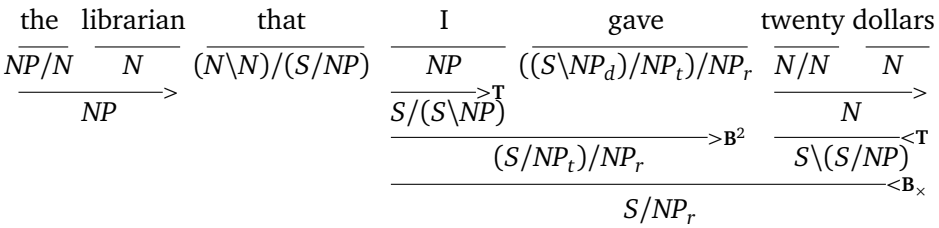
(4.3) I sold him a goldfish.

(4.4) *I sold a goldfish him.

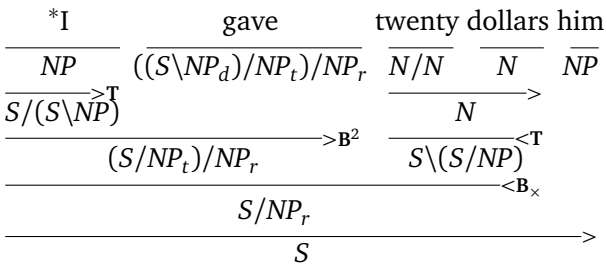
The subject argument of a ditransitive verb is known as the *donor*, while the two direct object arguments of ditransitive verbs are labelled the *theme* and *recipient*, where the theme is the entity being transferred from donor to recipient. Canonically, the recipient precedes the theme.

(4.5) Egbert$_{\textbf{Agent}}$ makes Hortense$_{\textbf{Recipient}}$ a treehouse$_{\textbf{Theme}}$.

To elucidate this fact, we annotate the direct object arguments of the ditransitive verb category as $((S\backslash NP_d)/NP_t)/NP_r$ to denote that they are not interchangeable. Now, consider the CCG derivation in Figure 4.5[1].



(a) Object extraction from ditransitive verb recipient with no overgeneration



(b) Overgeneration

Figure 4.5: An analysis of overgeneration with object extraction

The role of the modes $\ll$ and $\gg$ in preventing the overgeneration of Example 4.5b is that crossed composition, where the direction of the triangular arrowhead opposes that

[1]In this derivation, we have used a new combinatory rule of *generalised composition* as used by Steedman [2000], which generalises composition over categories which take multiple successive arguments. The rule of generalised forward harmonic composition which we have used in Examples 4.5a and 4.5b can be expressed as follows:

$$X/Y \quad (Y/Z_1)/Z_2 \quad \Rightarrow \quad (X/Z_1)/Z_2 \qquad (>\mathbf{B}^2) \tag{4.6}$$

The analogous rule of generalised backward crossed composition ($<\mathbf{B}^2_\times$) is also used in English.

of the directionality of the slash itself marks the result category for *antecedent government* ($\pm ANT$)[2]. A number of the crossed composition rules are made available when the input categories are marked for antecedent government, however after using these rules, the resulting category is also marked for antecedent government. In Baldridge's formulation, the only lexical category which expects an antecedent governed argument is the category of a subject or object-extracting relative pronoun, $(N\backslash_\star N)/_\star(S \mid NP)_{[+ANT]}$. In summary, antecedent government makes available a limited degree of crossed composition, by restricting the context in which it may be used to the body of a relative clause.

By marking a slash with a mode directionality ($\diamond\!\!\times$, $\times\!\!\diamond$) contrary to the directionality of its slash ($/, \backslash$), we can cause the use of backward crossed composition to induce antecedent government. A category marked $+ANT$, in fact, cannot serve as input to a combinatory rule whose argument is not marked for antecedent government.

There are now four versions of the backward crossed composition rule, one case for each combination of mode $\diamond\!\!\times$ or $\times\!\!\diamond$ on the left and right arguments to the rule[3].

$$Y/_{\times\!\diamond}Z \quad X\backslash_{\diamond\!\times}Y \quad \Rightarrow X/_{\times\!\diamond}Z \tag{4.7}$$

$$Y/_{\times\!\diamond}Z_{[+ANT]} \quad X\backslash_{\times\!\diamond}Y \quad \Rightarrow X/_{\times\!\diamond}Z_{[+ANT]} \tag{4.8}$$

$$Y/_{\diamond\!\times}Z_{[+ANT]} \quad X\backslash_{\diamond\!\times}Y \quad \Rightarrow X/_{\diamond\!\times}Z_{[+ANT]} \tag{4.9}$$

$$Y/_{\diamond\!\times}Z_{[+ANT]} \quad X\backslash_{\times\!\diamond}Y \quad \Rightarrow X/_{\diamond\!\times}Z_{[+ANT]} \tag{4.10}$$

Notice that the feature $+ANT$ is "contagious", in the sense that whenever the left hand argument carries the feature $+ANT$, then the result is also marked $+ANT$.

Figure 4.6 shows that antecedent government stops the overgeneration of Figure 4.5. In combining the category $(S/NP_t)/_\triangleleft NP_r$ with $S\backslash(S/NP)$, the result is marked for antecedent government because the directionality ($\triangleleft$) is contrary to that of the slash ($/$) in the first argument. Since the input categories to the rules of application are not marked for antecedent government, application does not succeed, blocking the overgeneration.

Since we have collapsed all of the modes permitting composition, harmonic or crossed, into a single mode $\circ$, we cannot make this grammaticality distinction with our simplified mode scheme. On the other hand, C&C currently does not mark antecedent government, so the distinction would be ignored in the parser. Furthermore, C&C allows at most one feature to be attached to a category as an efficiency measure and because

[2]Antecedent government is a relation between nodes defined in the linguistic theory of Government and Binding [Chomsky, 1993]. Simplifying the definition given, a node v is antecedent governed by u when v lies in the subtree of the sibling of u, and u and v are co-indexed (that is, u stands in for a constituent deleted from within v).

[3]Versions of the forward crossed composition rule ($> \mathbf{B}_\times$) which respect antecedent government are analogous to the rules for backward crossed composition given here. Just as forward crossed composition is generally not permitted in English (see Section 4.3.4), generalised forward crossed composition ($>\mathbf{B}_\times$) is also not active in English. The full set of rules is given by Baldridge [2002].

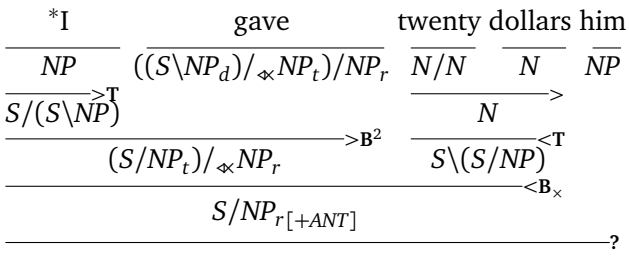


Figure 4.6: Modes $\ll$, $\gg$ preventing overgeneration

no more than one feature is attached to a category in CCGbank. Thus, to capture this analysis, we would either have to mark antecedent government orthogonally to the feature system, or else modify C&C categories to accept multiple features. To augment C&C to mark categories for antecedent government would require considerable structural changes. In this initial investigation of MMCCG, we omit the consideration of antecedent government, and hence the fine-grained modes $\ll$ and $\gg$ and the implementation of Rules 4.7 to 4.10, leaving their implementation in C&C for future work.

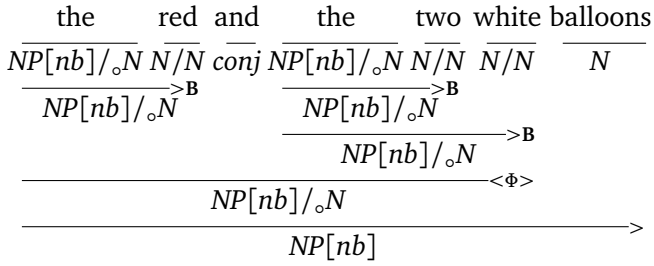
The second difference between our simplified mode scheme and the original as given by Baldridge, is the collapse of the harmonic composition-permitting mode ($\diamond$) and the family of crossed composition-permitting modes discussed above ($\ll$, $\gg$, $\triangleleft$, $\triangleright$, $\cdot$) into the single mode $\circ$. At the very least, in future work, we would like to distinguish between crossed and harmonic composition in even a simplified mode scheme. To see that there are circumstances in which harmonic but not crossed composition is required, consider the analysis in Figure 4.7. Ideally, in Baldridge's full scheme, *the* should carry category $NP[nb]/_{\diamond}N$, with the mode $\diamond$ permitting harmonic but not crossed composition, to admit Example 4.7a while blocking Example 4.7b.

The third change we have made to Baldridge's mode scheme is our novel addition of the *null mode* ($\bowtie$). The null mode on a slash is not licensed to participate in any combinatory rule, including the rules of application licensed by the most restrictive mode $\star$ in Baldridge's hierarchy. We will motivate the utility of this change in Section 4.3.2.

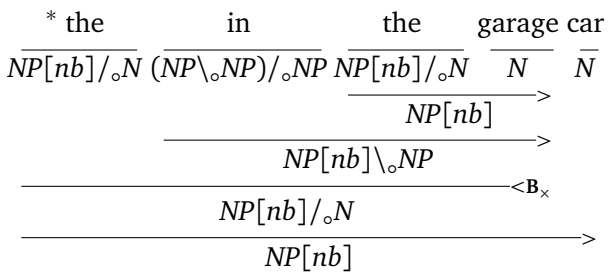
4.2.2 Differences in implementation

Although Baldridge's contribution is almost entirely an exposition of his theoretical extension to CCG, he performs an informal analysis of the practicability of MMCCG by extending a CCG parser, *Grok*, co-authored with Gann Bierner, to support his modifications. Grok is comprised of 11 000 lines of Java, and is designed with a high degree of modularity, contrary to the implementation philosophy of C&C.

Unlike C&C, Grok is not a wide-coverage parser, does not acquire a statistical model



(a) Harmonic composition is required for coordination



(b) Crossed composition leads to overgeneration

Figure 4.7: Inability to distinguish harmonic and crossed composition overgenerates

of language and thus is limited in practical use, and is designed with modularity and ease of modification over efficiency.

The emphasis on modularity, however, allows Grok to be used as a test-bed for the small-scale evaluation and analysis of language. While the goal of C&C is wide-coverage parsing of general text backed by a statistical model, Grok is designed so that researchers can easily make experimental changes to the base formalism, or to the input and output processing of the parser, and examine their impact, typically on a small, hand-crafted lexicon of several hundred lexical items.

It is this quality which allows Baldridge to make straightforward changes to the representation of categories to add modes to the slashes. Baldridge also describes that it would be a simple matter to structure Grok's treatment of modes to allow the mode hierarchy itself to be modified in a modular fashion. In contrast, we chose to represent modes as bitpacked values in category objects (described in Chapter 5), which reduces the space required to specify a mode, at the cost of generality.

Furthermore, Grok supports the concept of *mode variables* in the context of type-raising. The usual formulation of the MMCCG type-raising rule schemata are as follows:

$$X \quad \Rightarrow \quad T/_i(T\backslash_i X) \qquad (>\mathbf{T}) \qquad (4.11)$$

$$X \quad \Rightarrow \quad T\backslash_i(T/_i X) \qquad (<\mathbf{T}) \qquad (4.12)$$

The intended effect is that the newly generated slashes of a type-raised category can carry any mode as required by the derivation. To simulate this directly in Grok, Baldridge gives the slashes of the result of type-raising a mode variable i, which, intuitively, is a kind of "wildcard mode" which unifies as necessary when combined through subsequent applications of combinatory rules. In C&C, we choose to implement type-raising in a different way. Instead of employing a type variable, we simply give the slashes of the type-raised category the maximally permissive mode ($\circ$). Functionally, we have:

$$X \quad \Rightarrow \quad T/_\circ(T\backslash_\circ X) \qquad (>\mathbf{T}) \tag{4.13}$$

$$X \quad \Rightarrow \quad T\backslash_\circ(T/_\circ X) \qquad (<\mathbf{T}) \tag{4.14}$$

such that the type-raised category is compatible with any combinatory rule. As an example, consider Figure 4.8.

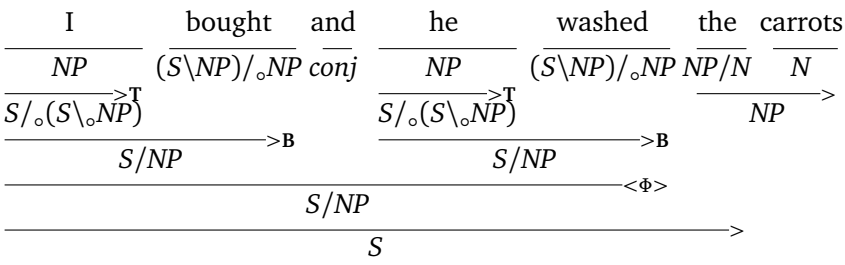


Figure 4.8: Our approach to the mode of a type-raised category

The type-raised category of *I* carries the maximally permissive mode ($\circ$), ensuring that it is compatible with any subsequent combinatory rule. There is one significant disadvantage of this scheme over the direct use of mode variables: suppose a given category X is frequently type-raised to $T/(T\backslash X)$ then subsequently used as an argument to functional application. A parser could then try the type-raise $T/_\star(T\backslash_\star X)$, rather than the more permissive $T/_\circ(T\backslash_\circ X)$. This would reduce parser ambiguity in the same way that any restrictive mode assignment does: by restricting the set of combinatory rules that need to be considered together with that category. Despite this, in our current implementation of mmccg through C&C, we choose to implement type-raising as per Rules 4.13 and 4.14 above, leaving a fuller model of type-raising for future work.

Having described the differences between our treatment of mmccg and the original formulation given in Baldridge [2002], we now describe the role of each of the three modes present in our reduced mode system, justifying their presence with reference to those syntactic constructions enabled by the power of each of the modes.

4.3 Characterising our mode scheme

4.3.1 Preliminaries

Definition 4. *The* canonical application-only order *of a category $X|Y$ is a sequence defined recursively as Y, followed by the canonical application-only order of X. The canonical application-only order of an atomic category C is itself.*

In the following sections, we will employ the above definition to distinguish the argument-taking order specified by the structure of the category itself, from an order modulated by combinatory rules. For example, the canonical application-only order of $((S\backslash NP_1)/NP_2)/NP_3$ is $\langle NP_3, NP_2, NP_1\rangle$, while the order in which its arguments are ultimately taken can differ in the presence of combinatory rules.

We now continue with the analysis of the three modes in our system, motivating their presence with reference to the syntax of English.

4.3.2 Null mode ($\bowtie$)

One of the contributions we introduce in this work is a new mode not present in Baldridge's mode scheme, motivated by our analysis into the distribution of the combinatory rules which consume each slash in CCGbank. A slash carrying the null mode ($\bowtie$) is not licensed to participate in *any* combinatory rule. This may seem counterintuitive: why can't we replace such a category with an atomic category, if it cannot participate in any rule? The null mode is motivated by the analysis of various compound categories in CCGbank, which never act on another category; they are always consumed as the sought category of another category. A derivation involving the lexical item $\textit{has} \vdash (S[pt]\backslash NP)/NP$, which exhibits this behaviour, is given in Figure 4.9. The only lexical items which *take* an argument $S[pt]\backslash NP$ are the various forms of the auxiliary verb *has*, while the only lexical items which *yield* the result $S[pt]\backslash NP$ are the past participle forms of English verbs (for example, *given*, *brought* and *written*).

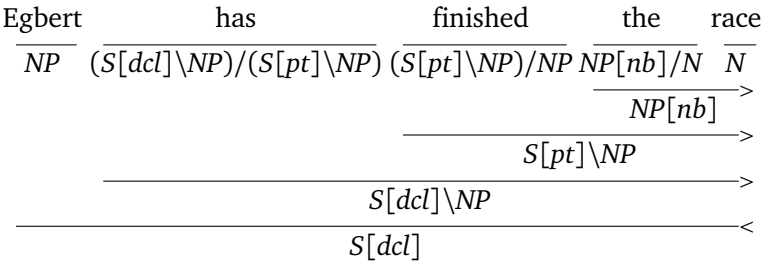


Figure 4.9: $S[pt]\backslash NP$ is only ever introduced by forms of *has*

Why can the category $S[pt]\backslash NP$ not be replaced by an atomic category? C&C is capable of generating *long-distance dependencies*: a word-word dependency which crosses a

constituent boundary. That is, we would expect to see in the dependency representation corresponding to Figure 4.9 the pair of dependencies shown in Figure 4.10.

i_1	i_2	c	r	l_1	l_2
⋮	⋮	⋮	⋮	⋮	⋮
0	2	$S[pt]\backslash NP)/NP$	1	Egbert	finished <XB>
0	1	$(S[dcl]\backslash NP)/(S[pt]\backslash NP)$	1	Egbert	has
⋮	⋮	⋮	⋮	⋮	⋮

Figure 4.10: Excerpt of dependency analysis for Figure 4.9

Thus, despite the fact that the category of *finished* never directly consumes its subject argument, the long-distance dependency between *finished* and *Egbert* is captured in the dependency analysis, marked with <XB>. If $S[pt]\backslash NP$ were replaced by an atomic category, then there would be no argument slot to be satisfied by the long-distance dependency.

The presence of categories which are never used in conjunction with any combinatory rule is required in order to capture dependencies such as the above. Regardless of this, such categories are still considered by C&C as candidates for combination, and such possibilities will be considered and added to the chart in the parsing process. The rationale for the null mode should now be clear: if we attach the null mode to any slash which has this property, then we can retain the dependency analysis while making more efficient use of parser resources.

OBSERVATION 1. *If the slash in $X|_i Y$ is not for direct combination with any rule, then it should receive the null mode ($\bowtie$).*

4.3.3 Application-only mode ($\star$)

A slash with the application-only mode ($\star$) is licensed only for functional application (Rules 2.10 and 2.11 in Figure 2.3). As described in Section 2.4.3, this mode is key to the MMCCG analysis of conjunction. More generally, the application-only mode serves to block the associative effects of harmonic composition and the permutative effects of crossed composition [Baldridge and Kruijff, 2004], allowing us to require that arguments be consumed strictly in the linear order in which they appear. This can be used to bound and limit constituent movement in constructions which allow a limited degree of it. We describe two classes of adverbs in English with distinct movement behaviour, one more restricted, and one with freer distribution.

VP adverbs are distinguished in English by their relatively free distribution within the verb phrase which they modify. Members of this class include *safely*, *now*, *often*,

quickly and *constantly*. VP adverbs regularly specify the manner or time in which an action is performed. Backward crossed composition allows the analysis of parts of speech which enjoy a degree of free word order with respect to other constituents, including the VP adverbs, as demonstrated by the position of the adverb *freely* in Examples 4.15–4.17. In CCG, such adverbs receive one of the categories $(S\backslash NP)\backslash(S\backslash NP)$ or $(S\backslash NP)/(S\backslash NP)$, depending on whether their canonical position (the position which yields an application-only analysis) is *preceding* the verb phrase *permute within their phrase* (as in Example 4.16), or *following* it (as in Example 4.17). The rules of composition allow for the analysis of an adverb shifted from either of these canonical positions, as in Example 4.15.

(4.15) Adverbs permute *freely* within their phrase.

(4.16) Adverbs *freely* permute within their phrase.

(4.17) Adverbs permute within their phrase *freely*.

However, English and a number of other languages exhibit a class of *sentential adverbs*, the distribution and syntax of which differs markedly from the *VP adverbs* described above. While VP adverbs often express the manner in which an action is performed, a sentential adverb often encodes the speaker's judgement or attitude towards the utterance [Astuc-Aguilera and Nolan, 2007]. Example members are *clearly*, *obviously*, *unfortunately*, *doubtless*, *probably* and *surely*. Semantically, sentential adverbs modify the sentence, as opposed to the verb phrase. However, they have a hybrid distribution in that they can appear in one of three places: preceding the verb phrase (Example 4.18), or either preceding (Example 4.19) or following (Example 4.20) the basic sentence, offset by a comma. Examples 4.22 and 4.21 show that sentential adverbs do not follow the distribution of the VP adverbs.

(4.18) He evidently knows some judo.

(4.19) Evidently, he knows some judo.

(4.20) He knows some judo, evidently.

(4.21) He knows some judo evidently.

(4.22) *He knows evidently some judo.

In CCGbank, the occurrences of *evidently* in Examples 4.19 and 4.20 would receive the sentence modifier categories S/S and $S\backslash S$ respectively. However, in Example 4.18 it would receive the category $(S\backslash NP)/(S\backslash NP)$, identical to that of a VP adverb. This will, however, overgenerate the ungrammatical Example 4.22 in the same way that it generates the grammatical Example 4.15. Accordingly, a parser trained on CCGbank cannot discriminate between VP adverbs and the movement-restricted sentential adverbs.

However, with multi-modal CCG we can prevent the overgeneration of Example 4.22 by placing the application-only mode ($\star$) on the first slash of the category of a sentential

adverb (that is, *evidently* $\vdash (S\backslash NP)/_{\star}(S\backslash NP)$) to prevent it from undergoing the degree of free movement available to a VP adverb.

Generally, the application-only mode is useful in restricting the ability of a constituent to permute, by confining it to forward and backward application, enabling us to encode new grammaticality distinctions. We formulate this in the following way:

Observation 2. *If no other constituent may intervene between category $X|_i Y$ and its argument category Y, then the slash should receive the application-only mode ($\star$).*

4.3.4 Maximally permissive mode ($\circ$)

The maximally permissive mode ($\circ$) enables all of the combinatory rules, although in English, the distribution of most of the rules provided by CCG is either extremely limited, or non-existent. Substitution is only involved in the analysis of the *parasitic gap* construction, which calls upon its facility of allowing a category to serve simultaneously as an argument to two categories in a limited way [Steedman, 2000].

(4.23) Which cake did you touch without eating?

This construction is known as a parasitic gap, because it involves *two* deletions, one of which is said to be parasitic on the other. Making clear the positions from which the extractions occur, we have:

(4.24) Which cake did you touch x_i without eating x_i?

where x_i represents the extracted constituent *cake*. Hockenmaier [2003] deemed the distribution of parasitic gaps in too uncommon enough to justify their treatment in CCGbank, and consequently the implementation of the rules of substitution is absent from C&C.

Forward crossed composition is typically entirely de-licensed for English (in C&C, the rule is left unimplemented for performance reasons), because its unchecked application leads to *scrambling*, or the complete breakdown of word order in a non-free word order language. This is known as *permutation closure*, since it implies that if a given string is accepted, then any permutation of its words is also accepted. Although uncontrolled scrambling is certainly undesirable in languages with rigid word order, it is unsurprising that forward crossed composition is useful in analysing languages with freer word order than English, such as Turkish [Baldridge, 2002] or Latin.

Nevertheless, the remaining forms of composition: forward and backward harmonic ($>\mathbf{B}, <\mathbf{B}$), and backward crossed ($<\mathbf{B}_{\times}$) are key to the analysis of a range of common syntactic structures in English. We observe in the following sections that the composition licensed by the maximally permissive mode is necessary in English to analyse syntax which involves constituent shift and gapping (deletion of constituents). This is evidence

$$\dfrac{\text{encouraged} \quad \dfrac{\text{Egbert}}{NP}\,{>}\mathbf{T}\ \dfrac{\text{him}}{NP}\,{<}\mathbf{T}}{\dfrac{(S\backslash NP)/NP \quad S/(S\backslash NP) \quad (S\backslash NP)\backslash((S\backslash NP)/NP)}{\dfrac{S\backslash((S\backslash NP)/NP)}{S}\,{<}}\,{>}\mathbf{B}_{\times}}$$

Figure 4.11: Scrambling caused by forward crossed composition

that the categories involved in these structures must carry appropriate modes on their slashes in order to permit these analyses.

4.3.4.1 Argument cluster coordination

Argument cluster coordination arises when the category of a conjunct must consume arguments in an order different to that specified by the canonical application-only order of its category structure. The derivation in Figure 4.12a does not require the use of composition, because each of the conjuncts consumes all of its arguments in canonical, application-only order, before conjoining. However, consider Figure 4.12b. The category structure of the verb *sat* $\vdash (S[dcl]\backslash NP)/PP$ indicates that the canonical order in which it takes its arguments is *PP* to the right, then *NP* to the left. However, in order to conjoin, it must consume arguments in a different order: either by combining with a type-raised *NP* to the left, and a particle PP/NP to the right, or alternatively in the reverse order[4]. The resulting conjunct category, S/NP, agrees with our intuition: it forms a sentence once it is combined with an object on the right.

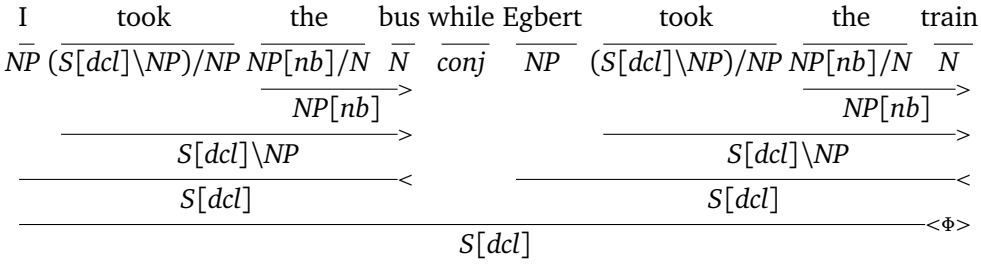
The judicious use of forward type-raising ($>\mathbf{T}$) followed by forward or backward harmonic composition ($>\mathbf{B}$, $<\mathbf{B}$) allows the analysis of coordination cases such as the above. Considering the multi-modal CCG modality annotation required in order to preserve complex coordination analyses, we give the following observation.

Observation 3. *For a category $X|_i Y$ to consume arguments in an order different from the canonical application-only order, the slash must be licensed for composition.*

4.3.4.2 Subject/object extraction

A relative clause is an embedded sentence introduced by one of the *relative pronouns*: in English, *who*, *whom*, *which*, *that*, and *whose*, together with a *null morph* (as in Example 4.27, a relative clause introduced by juxtaposition) which introduces a *reduced relative clause* [Quirk et al., 1985]. A relative clause is either *restrictive*, shrinking the set

[4]We can confirm that the resulting category is the same in both cases; this is an instance of spurious ambiguity, as demonstrated earlier in Section 2.2.4.



(a) Coordination not requiring >**B**

(b) Coordination requiring >**T** and >**B**

Figure 4.12: Forms of coordination

of potential referents by providing more information, or *non-restrictive*, giving incidental information about the referent. We can distinguish the two by erasing the relative clause and seeing if the entity that the relative clause modifies is made any less specific. If this is the case, the relative clause is restrictive, otherwise it is not. In the below examples, only Example 4.28 is non-restrictive, because we can erase the relative clause *who won two awards* without changing the referent.

(4.25) Here is the item *which* you ordered.

(4.26) The door *that* creaks is in another room.

(4.27) I gave him the two corndogs I was holding.

(4.28) The long-time director, who won two awards, is "satisfied with the accolade."

A relative pronoun has co-reference with (*refers back to*) an antecedent, just like the familiar personal pronouns. In the first case below, the personal pronoun *he* co-refers to *Egbert* in the same way that the relative pronoun *that* has co-reference with *the person* in the second.

(4.29) *Egbert* broke the vase. He didn't admit to it.

(4.30) They are looking for *the person* who broke the vase.

Uniquely, however, the relative pronouns are said to exhibit *subject or object extraction*, by which the co-referent within the relative clause is obligatorily deleted. Subject and object extraction differ by the grammatical role (subject or object) of the co-referent being deleted. The below examples exhibit extraction, where we mark the original position of the co-referent with ∅. We also see that extraction can succeed within a prepositional phrase (Example 4.35).

(4.31) They are looking for the person who ∅ broke the vase.
(4.32) *They are looking for the person who he broke the vase.
(4.33) Did you see the cable-car that the robot hit ∅?
(4.34) *Did you see the cable-car that the robot hit it?
(4.35) Who is the man that you gave it to ∅?
(4.36) *Who is the man that you gave it to him?

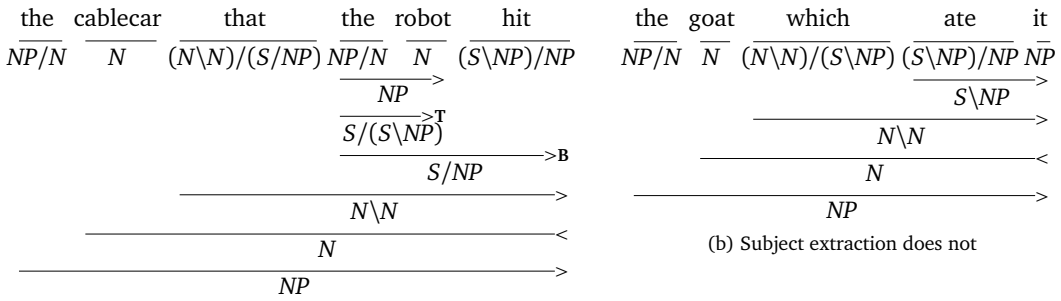
In a typical English CCG analysis, the category of a relative pronoun differs depending on whether it is engaged in subject or object extraction. The category of an object-extracting relative pronoun is $(N\backslash N)/(S[dcl]/NP)$: its argument is that of a sentence missing its object $(S[dcl]/NP)$, and its result is the same as the category of an noun post-modifier ($N\backslash N$, cf. the category of an adjective N/N, which is essentially a noun pre-modifier)[5]. Similarly, the category of an subject-extracting relative pronoun is $(N\backslash N)/(S[dcl]\backslash NP)$.

We now show that object extraction requires the use of composition, and hence that any verb capable of object extraction must carry mode $\circ$ on one of its slashes. Fundamentally, composition is necessary for the same reason that it is needed for argument cluster coordination in Section 4.3.4.1: to consume arguments in an order different to the canonical application-only order. Considering the category of a transitive verb $(S[dcl]\backslash NP_1)/NP_2$, it is clear that in order to perform object extraction, we need to fulfill the subject argument (NP_1) while leaving the object argument (NP_2) unconsumed. By contrast, subject extraction does not require composition, because the canonical application-only order consumes the object first, such that we can obtain the required subject-extracted category $S\backslash NP$ with application only.

Thus, in order to participate in object extraction, any verb category must have the mode ($\circ$) on the slash which consumes an object argument. For example, the category of a transitive verb must be $(S[dcl]\backslash NP)/_{\circ}NP$.

[5]Indeed an adjective is semantically similar to a relative clause, in that it either restricts, or further describes a noun (or noun phrase).

(4.37) I read the book *that had a woodcut of a parrot on the cover.*
(4.38) I read the *purple* book.



(a) Object extraction requires composition

(b) Subject extraction does not

Figure 4.13: Subject and object extraction

Observation 4. *If the object argument of a verb is a valid target for extraction, then the slash which consumes the object argument must carry a mode compatible with harmonic composition.*

4.3.4.3 Constituent shift: adverbials in English

Adverbials are a class of clauses with a modificatory or specificatory semantic role in a sentence. Adverbials are *not* a part of speech like the similarly named adverbs[6], but rather a grammatical role in a sentence, like subject or object.

(4.41) The suitcase remains the property of the Debt Recovery Office *until the third of March.*

(4.42) He walked *all the way from Glamorgan* to Glyncorrwg.

(4.43) *As far as I am concerned*, it is no longer a problem.

Quirk et al. [1985] distinguishes the adverbials by two qualities: their "range of possible positions in the clause", and their "propensity for multiple occurrence in the same clause". In this section, we will show that it is precisely the analysis of these two qualities which require the permutative power of the combinatory rule of backward crossed composition. To see this, observe that each of the following sentences is also grammatical:

(4.44) The suitcase, *until the third of March*, remains the property of the Debt Recovery Office.

(4.45) He walked to Glyncorrwg *all the way from Glamorgan*.

[6]Quirk et al. [1985] defines adverbs as modifiers of adjectives, as in Example 4.39 or otherwise of other adverbs, as in Example 4.40. Adverbials, in contrast, modify *verb phrases*, and are very broad in their syntactic distribution.

(4.39) That was a *remarkably* dull rendition of the Eroica.

(4.40) She was *always so very* kind.

(4.46) It is no longer a problem *as far as I am concerned*.

Adverbials receive the category $(S\backslash NP)\backslash(S\backslash NP)$ or $(S\backslash NP)/(S\backslash NP)$, depending on whether they follow, or precede the verb phrase that they modify. Observe the range of positions which the adverbial may take:

(4.47) She $\textit{stubbornly}_{(S\backslash NP)/(S\backslash NP)}$ continues to refuse the office of president.

(4.48) She continues $\textit{stubbornly}_{(S\backslash NP)\backslash(S\backslash NP)}$ to refuse the office of president.

(4.49) She continues to $\textit{stubbornly}_{(S\backslash NP)/(S\backslash NP)}$ refuse the office of president.

(4.50) She continues to refuse $\textit{stubbornly}_{(S\backslash NP)\backslash(S\backslash NP)}$ the office of president.

(4.51) She continues to refuse the office of president $\textit{stubbornly}_{(S\backslash NP)\backslash(S\backslash NP)}$.

Although Examples 4.47 and 4.51 are the canonical application-only positions for categories $(S\backslash NP)/(S\backslash NP)$ and $(S\backslash NP)\backslash(S\backslash NP)$ respectively, the remaining cases feature the adverbial being shifted from these positions.

Constituent shift of adverbials is handled in English either by *forward harmonic composition* ($>\mathbf{B}$) or *backward crossed composition* ($<\mathbf{B}_\times$), depending on which of the two below cases is satisfied:

(1) If we are moving an adverbial which would canonically *precede* the verb phrase to the *right*, we use forward harmonic composition ($>\mathbf{B}$)

(2) We are moving an adverbial which would canonically *follow* the verb phrase to the *left*, we use backward crossed composition ($<\mathbf{B}_\times$)

The derivations for Examples 4.47–4.51 are given in Figure 4.14. We have heavily condensed the derivations except for the step in which the adverbial is combined.

We have shown that in order to analyse adverbial shift in English, two of the rules of composition are required. Indeed, the analysis of free movement to the left and right in *any* language requires the more powerful combinatory rules of composition, and hence the category of a constituent which undergoes free movement must carry permissive modes on its top level slash.

Observation 5. *Adverbials in English carry the* MMCCG *categories* $(S\backslash NP)/_\circ(S\backslash NP)$ *and* $(S\backslash NP)\backslash_\circ(S\backslash NP)$*, to permit the analysis of their free movement within a verb phrase.*

$$\dfrac{\text{She}\quad \dfrac{\text{stubbornly}}{(S\backslash NP)/(S\backslash NP)}\quad \dfrac{\text{continues to refuse the office of president}}{S\backslash NP}}{S\backslash NP}{>}$$

(a)

$$\dfrac{\text{She}\quad \dfrac{\text{continues}}{(S\backslash NP)/S[to]}\quad \dfrac{\text{stubbornly}}{(S\backslash NP)\backslash(S\backslash NP)}\quad \dfrac{\text{to refuse the office of president}}{S[to]}}{(S\backslash NP)/S[to]}{<\mathbf{B}_{\times}}$$

(b)

$$\dfrac{\text{She}\quad \dfrac{\text{continues}}{(S\backslash NP)/S[to]}\quad \dfrac{\text{to}}{S[to]/(S[b]\backslash NP)}\quad \dfrac{\text{stubbornly}}{(S\backslash NP)/(S\backslash NP)}\quad \dfrac{\text{refuse}}{(S[b]\backslash NP)/NP}\quad \dfrac{\text{the office of president}}{NP}}{(S\backslash NP)/NP}{>\mathbf{B}}$$

(c)

$$\dfrac{\text{She}\quad \dfrac{\text{continues}}{(S\backslash NP)/S[to]}\quad \dfrac{\text{to}}{S[to]/(S[b]\backslash NP)}\quad \dfrac{\text{refuse}}{(S[b]\backslash NP)/NP}\quad \dfrac{\text{stubbornly}}{(S\backslash NP)\backslash(S\backslash NP)}\quad \dfrac{\text{the office of president}}{NP}}{(S\backslash NP)/NP}{<\mathbf{B}_{\times}}$$

(d)

$$\dfrac{\text{She}\quad \dfrac{\text{continues to refuse the office of president}}{S\backslash NP}\quad \dfrac{\text{stubbornly}}{(S\backslash NP)\backslash(S\backslash NP)}}{S\backslash NP}{<}$$

(e)

Figure 4.14: CCG analyses for adverbial shift in each of its possible positions

4.3.4.4 Summary

We have shown that composition is key to capturing complex syntactic phenomena in English, with crossed composition enabling a degree of permutativity and hence freedom of word order, while harmonic composition allows the arguments of a category to be consumed contrary to the canonical application-only order. Each of the three modes corresponds to a useful degree of freedom in the mode hierarchy: the null mode ($\bowtie$) allows us to preserve the structure of a category for dependency analysis while preventing the parser from considering it as an argument to any combinatory rule. The application-only mode ($\star$) allows us to selectively restrict the power of MMCCG to prevent overgeneration caused by the more powerful rules of composition, and finally the maximally permissive mode ($\circ$) makes available the full spectrum of combinatory rules to maintain the powerful CCG analyses for which it is known.

4.4 Approaches to deriving MMCCG corpora

Section 4.3.4 is an inventory of the syntactic constructions enabled by permitting the more powerful combinatory rules of composition. The ultimate goal of this work is to limit the parser's consideration of these rules to categories involved in only the above cases, with two intertwined goals: to increase the parser's discriminative power by preventing them from applying where they do not yield a meaningful analysis, and to increase the parser's efficiency by considering them only when they may yield a meaningful analysis.

An entirely automatic corpus conversion process has three main benefits over one that involves manual annotation. Firstly, automatic corpus conversion allows for the possibility of a language-agnostic procedure which allows us to generate a MMCCG corpus for any language for which we have a CCG corpus. Secondly, by requiring no human annotation, we greatly reduce the cost in time and human resources allocated to the manual process. On the other hand, we have seen that it is difficult to make mode distinctions solely on the basis of frequency, because relatively uncommon syntactic constructions can still be entirely grammatical. Lastly, the implementation of the annotation procedure directly documents our decisions, unlike manual annotation, which is generally an inscrutable process by which annotators make judgements without providing a case-by-case justification. On the other hand, additional noise may be introduced into the corpus as a result of automated annotation, a case of *garbage in, garbage out*. On the other hand, noise introduced by human annotators is less consistent, but rarer.

Manual annotation is precise, but requires considerable time and effort. While the frequency-based automatic process described above may introduce noise by assigning overly restrictive modes, a human annotator can use their own grammaticality intuition to disregard noisy analyses and also generalise a pattern of mode assignments past what is observed in the source corpus. For example, if an annotator notices that the slash which consumes the subject of a transitive verb requires a particular mode, they may be able to extend that analysis to other classes of verbs which must also support such an analysis. However, annotating each slash of each category observed in CCGbank is a mammoth undertaking, especially with the presence of noise, and the occurrence of very rare or very complex categories.

Our methodology combines automatic and manual annotation to achieve a compromise between the considerable effort of annotating categories with modes by hand, and the imprecision of performing the annotation on statistical grounds. In particular, the assignment of the null ($\triangleright\!\!\triangleleft$) and application-only modes ($\star$) can be performed automatically, while we rely on manual annotation to distinguish the need for composition ($\circ$), a task which puts human grammaticality judgement to good use.

4.4.1 Corpus analysis

In attempting to determine an appropriate way to attack the annotation problem, we developed corpus exploration tools which allowed us to consider the frequency with which each slash of each category was *consumed* by each combinatory rule in CCGbank. We define a slash i of a category C as having been consumed by a combinatory rule, if that rule caused the slash to vanish, or merge with another slash. Since type-raising actually creates slashes rather than causes them to disappear or be merged, it is not considered to be a consuming rule. We define each of the rules in Figure 4.15 to *consume* slashes i (and j where present) of their arguments[7].

$$
\begin{array}{rrcll}
X/^{i}Y & Y & \Rightarrow & X & (>) \\
Y & X\backslash^{i}Y & \Rightarrow & X & (<) \\
X/^{i}Y & Y/^{j}Z & \Rightarrow & X/Z & (>\mathbf{B}) \\
Y\backslash^{i}Z & X\backslash^{j}Y & \Rightarrow & X\backslash Z & (<\mathbf{B}) \\
X/^{i}Y & Y\backslash^{j}Z & \Rightarrow & X\backslash Z & (>\mathbf{B}_{\times}) \\
Y/^{i}Z & X\backslash^{i}Y & \Rightarrow & X/Z & (<\mathbf{B}_{\times})
\end{array}
$$

Figure 4.15: Defining which combinatory rules *consume* their slashes

This analysis allows us to determine how many times each slash of each category has been consumed by each combinatory rule. Carrying out this analysis for every slash of every category in CCGbank, we obtain data similar to Table 4.2[8]. We will call each row in this analysis a *slash/combinator pair*, because it identifies the number of times which a particular slash belonging to a category occurs with a particular combinatory rule. Table 4.1 additionally shows the frequency with which each of the above rules consumes each slash present in the category of a leaf in CCGbank[9]. We can explain the distribution of slashes seen in Table 4.1 with reference to English syntax. The prevalence of forward application (>) over backward application (<) reflects the *right-branching* tendency of English (although Example 4.16c demonstrates that branching is indeed just

[7]For completeness, we also note that the input categories to the rules of substitution are considered to consume their top level slashes. These rules do not appear in Figure 4.15 because of their rarity of occurrence in English.

[8]All of the slash indices depicted in this section follow our ordering convention on slashes, given by Convention 2 of page 19).

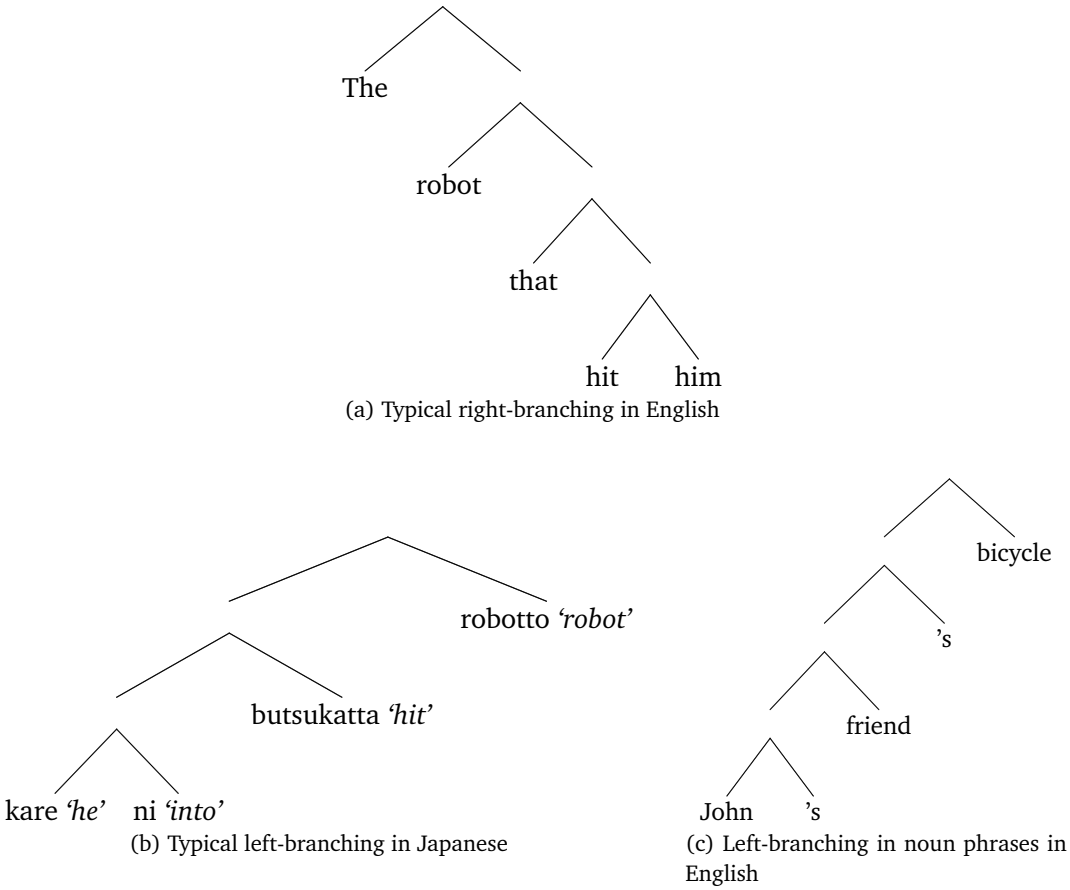
[9]The cell of Table 4.1 with no combinator indicates the number of slashes which were not directly consumed. For example, in the forward application:

$$S[dcl]/(S[adj]\backslash NP) \quad S[adj]\backslash NP \quad \Rightarrow \quad S[dcl]$$

the slash in the argument of the first input category is never consumed directly.

a tendency). A typical CCG analysis of a right-branching language would bias forward application over backward application.

We also see from the table that backward crossed, and forward harmonic composition, which appear in our analysis in Section 4.3.4.3 of rightward and leftward movement of adverbials, are relatively rare, with the rarest consuming rule being backward harmonic composition ($<\mathbf{B}$), which allows the analysis of argument cluster coordination shown in Figure 2.9.



(a) Typical right-branching in English

(b) Typical left-branching in Japanese

(c) Left-branching in noun phrases in English

Figure 4.16: Left- and right-branching in Japanese and English

Combinator	*Frequency*	
$>$	656471	(49.47%)
	438977	(33.08%)
$<$	197893	(14.91%)
$<\mathbf{B}_{\times}$	26388	(1.99%)
$>\mathbf{B}$	6714	(0.51%)
$<\mathbf{B}$	517	(0.04%)

Table 4.1: Frequency of consumption by combinatory rule in CCGbank

	Category	*Slash index*	*Combinator*	*Frequency*
1	N/N	0	$>$	137176
2	$NP[nb]/N$	0	$>$	77951
3	$(NP\backslash NP)/NP$	0	$>$	41064
4	$(NP\backslash NP)/NP$	1	$<$	39520
5	$((S\backslash NP)\backslash(S\backslash NP))/NP$	3		21112
6	$((S\backslash NP)\backslash(S\backslash NP))/NP$	0	$>$	21064
7	$((S\backslash NP)\backslash(S\backslash NP))/NP$	1	$<$	18633
8	$((S\backslash NP)\backslash(S\backslash NP))/NP$	2		16932
9	PP/NP	0	$>$	15827
10	$(S\backslash NP)\backslash(S\backslash NP)$	2		15307
11	$(S[dcl]\backslash NP)/NP$	0	$>$	13657
12	$(S[b]\backslash NP)/NP$	1		12213
13	$(S[to]\backslash NP)/(S[b]\backslash NP)$	2		12023
14	$(S[to]\backslash NP)/(S[b]\backslash NP)$	1		12022
15	$(S[to]\backslash NP)/(S[b]\backslash NP)$	0	$>$	11596
16	$(S[b]\backslash NP)/NP$	0	$>$	10862
17	$(S\backslash NP)\backslash(S\backslash NP)$	1		10196
18	$(S[dcl]\backslash NP)/(S[b]\backslash NP)$	2		10121
19	$(S[dcl]\backslash NP)/NP$	1		8530
20	$(S\backslash NP)\backslash(S\backslash NP)$	0	$<\mathbf{B}_{\times}$	8251
⋮	⋮	⋮	⋮	⋮

Table 4.2: Top 20 slashes from slash/combinator frequency analysis

With this data, we were able to make a number of observations, which suggest and support the approach that we formulate in Section 4.4. We determined two factors which influence our chosen approach:

Sparseness of distribution across slashes: The distribution of the frequency of slashes consumed by each combinatory rule is heavily long tailed. Also, composition cases are much rarer than application cases across all categories.

Sparseness of distribution within a slash: If we consider only a single slash, and consider the frequency with which each combinatory rule has consumed that slash, it is often the case that composition accounts for a miniscule proportion of the total, even when manual inspection confirms that the composition cases correspond to legitimate analyses.

Table 4.3, which aggregates the frequency with which each slash/combinator pair has been seen, clarifies the first factor. We see that although only 207/2722 (7.6%) slash/combinator pairs are seen more than 500 times in CCGbank, accounting for just these 207 slashes means that we will have annotated the categories of 73579 of the 79660 unique tokens (92.37%) occurring in the corpus. This demonstrates that annotating a small number of slashes will nevertheless allow us to cover a great deal of CCGbank.

Table 4.4 shows the frequency with which each combinatory rule has consumed the first slash of $NP[nb]/N$, the category of a determiner (*the*, *a*) in CCGbank[10]. Overwhelmingly, the determiner category is combined 99.47% of the time with simple forward application with a noun, to form a non-bare noun phrase, as in Figure 4.17a. However, in 354 cases (0.45%), the first slash of $NP[nb]/N$ is instead consumed through forward composition, typically in a coordination construction such as Example 4.17b. In English, such constructions are valid, although very rare compared to the straightforward case of Example 4.17a. The implication is that we cannot use a simple frequency cutoff to decide whether a given slash should receive a mode, solely based on the number of times it has been consumed by a given combinatory rule.

However, the fact that a small proportion of cases legitimately require composition has motivated us to mark the entire category for composition: $NP[nb]/_{\circ}N$, negating the efficiency benefits of assigning less permissive modes. Later in Section 4.5, we further develop our corpus derivation methodology to alleviate this problem.

We now proceed to describe the annotation approach we have developed specifically to deal with these two attributes of CCGbank.

[10]The row of Table 4.4 with no combinator filled in denotes the number of times the category was "swallowed whole" as the sought argument of another category (for example $NP/(NP[nb]/N)$), such that the slash it contains was never directly consumed by any combinatory rule.

Frequency range	*Count*
137000 – 137999	1
77000 – 77999	1
41000 – 41999	1
39000 – 39999	1
21000 – 21999	2
18000 – 18999	1
16000 – 16999	1
15000 – 15999	2
13000 – 13999	1
12000 – 12999	3
11000 – 11999	1
10000 – 10999	3
8000 – 8999	2
7000 – 7999	5
6000 – 6999	3
5000 – 5999	8
4000 – 4999	15
3000 – 3999	16
2000 – 2999	18
1000 – 1999	54
0 – 999	2583

Table 4.3: Number of slash/combinator pairs seen in each frequency band

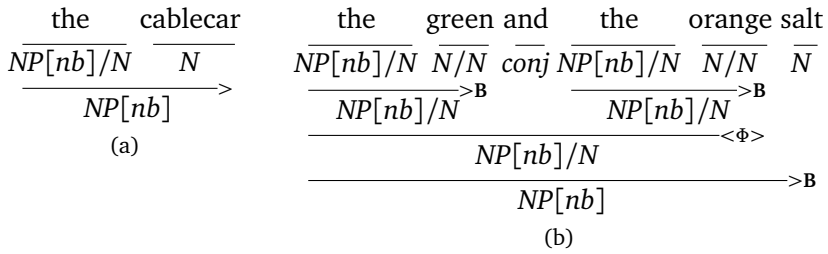


Figure 4.17: Application and coordination analyses involving $NP[nb]/N$

4.4.2 Automatic corpus annotation algorithm

We perform automatic annotation to attach the application-only ($\star$) and null ($\bowtie$) modes, either because they occur frequently enough that noise is not problematic (as is the case for the application-only mode), or otherwise because the evidence in the corpus for attaching the mode is so strong that it can be done reliably in an automated fashion (as is true for the null mode). We parameterise on a threshold value α: the relative frequency with which a given slash must be observed to be consumed in CCGbank. For

Comb.	*Frequency*	
>	77951/78366	(99.47%)
>B	354/78366	(0.45%)
	61/78366	(0.08%)

Table 4.4: Distribution of the combinatory rules which consumed slash 0 of $NP[nb]/N$

example, if a slash is consumed by application 99% of the time in CCGbank, and our threshold is $\alpha = 98\%$, then the algorithm would assign it the application-only mode $\star$ in the new corpus. In the one-to-one mode mapped corpus generated in this section, we use $\alpha = 0.95$, which is sufficient to prevent noise in combinatory rules from affecting our mode assignment.

The complete algorithm implementing automatic annotation is given in Figure 4.18.

1: **for each** category C occurring in CCGbank **do**
2: **for each** slash i of category C **do**
3: **if** the frequency with which slash i is consumed by application $> \alpha$ **then**
4: give slash i the mode $\star$ in the new corpus
5: **else if** the frequency with which slash i is never consumed $> \alpha$ **then**
6: give slash i the mode $\bowtie$ in the new corpus
7: **end if**
8: **end for**
9: **end for**

Figure 4.18: Automatic threshold-based annotation for modes $\star$ and $\bowtie$

4.4.3 Manual annotation

We employ manual annotation to determine where to attach the maximally permissive mode ($\circ$). Manual annotation is motivated by the fact that the spectrum of syntactic structures enabled by the availability of the rules of composition, as discussed in Section 4.3.4, is hard enough for humans to identify, let alone for an algorithm to distinguish automatically short of parsing instances of the structures themselves. Accordingly, we have conducted a manual annotation of 63 slashes covering 59 distinct categories. We use as a guideline the analysis we carried out in Section 4.3.4, which identifies those syntactic structures in English whose analysis requires the additional power of composition. If we deem that a category validly participates in one of these structures, then we give its slashes the modes necessary to enable such an analysis. This is the power of manual annotation: although a linguist can instantly determine

the validity of an analysis, the use of automated means of identifying syntax risks contributing further noise to the corpus.

In annotating the corpus, we employ a greedy methodology, by considering slashes in descending order of the frequency with which they are consumed by a rule of composition. In doing so, we maximise the coverage improved with each new slash annotated.

4.4.4 Applying the annotation to CCGbank

Once we have obtained the sets of annotations yielded by both the automatic annotation of Section 4.4.2 and the manual annotation of Section 4.4.3, we must apply the substitution to CCGbank to yield a moded corpus. To see that this is not as simple as substituting the categories in the leaves of CCGbank derivations with their moded replacements, consider that the output categories of combinatory rules often copy the arguments or results of their input categories. For example, the rule of forward harmonic composition:

$$X/Y \quad Y/Z \quad \Longrightarrow \quad X/Z \qquad\qquad (>\mathbf{B})$$

copies the result X of its first, and the argument Z of its second input categories into the output category X/Z. If X, Y or Z are compound categories and contain modes, then it does not suffice to simply replace the leaf categories with their moded alternatives: we must *percolate* the mode changes up the derivation after performing the substitutions on its leaf categories, so that the resulting derivation tree is correct with respect to mmccg. This suggests that all we must do is recursively copy the modes on the slashes of the input categories to the corresponding parts of the output category. In the above example, we would recursively copy the modes from X and Z of the input categories into the output category.

However, there is a second complication: when the two input categories are compatible (according to the mode inheritance hierarchy) but not exactly the same, the slash of the output category should carry the less permissive of the modes on the slashes of the two input categories. As an example, given the two input categories with modes from Baldridge's full mode scheme:

$$X/_{\diamond}Y \quad Y/_{\times}Z \quad \Longrightarrow \quad X/_{\diamond}Z$$

the output category has mode $\diamond$ on its topmost slash, because $\diamond$ is the more restrictive of the two modes $\{\diamond, \times\}$ according to the mode hierarchy.

This means that we cannot percolate modes up the derivation tree while only considering the categories lying on a leaf-root path, because given any node in the derivation, the mode received by the *parent* slash depends on the mode on the *sibling* slash.

To address this dependency, we must therefore consider the nodes in *level-order*, such that before considering the categories at depth k, all of the categories at depth $k+1$ in the tree have already received their modes by percolation. The full algorithm implementing the single mode annotation and percolation scheme is given in Figure 4.19. In the algorithm, COPY-MODES(A,B) recursively copies the modes from A to B. COPY-MODES assumes that A and B share a structural category. Also, we define utility functions RESULT-OF and ARGUMENT-OF which respectively yield the result, or argument of a compound category. That is, RESULT-OF$(X|Y) = X$, while ARGUMENT-OF$(X|Y) = Y$.

```
LEVEL-ORDER-PAIRS(D):
{D is a sequence of node-sibling pairs from each leaf to the root}
Q ← ⟨D, nil⟩
result ← ⟨⟩
while not EMPTY?(Q) do
    cur_1, cur_2 ← PEEK(Q)
    APPEND(DEQUEUE(Q), result)
    if cur_2 ≠ nil and LEAF?(cur_2) then
        ENQUEUE(LEFT-CHILD(cur_2), RIGHT-CHILD(cur_2)
    end if
    if cur_1 ≠ nil and LEAF?(cur_1) then
        ENQUEUE(LEFT-CHILD(cur_1), RIGHT-CHILD(cur_1)
    end if
end while
return REVERSED(result)
```

```
PERCOLATE(D):
for each (left, right) ∈ LEVEL-ORDER-PAIRS(D) do
    parent ← PARENT-OF(left) {same as PARENT-OF(right)}
    {ANALYSE determines the combinator applied to left and right to get parent}
    comb ← ANALYSE(left, right, parent)
    if comb = (>B) or comb = (>B_×) then
        {X|Y   Y|Z → X|Z}
        COPY-MODES(RESULT-OF(left), RESULT-OF(parent))
        COPY-MODES(ARGUMENT-OF(right), ARGUMENT-OF(parent))
    else if comb = (<B) or comb = (<B_×) then
        {Y|Z   X|Y → X|Z}
        COPY-MODES(RESULT-OF(right), RESULT-OF(parent))
        COPY-MODES(ARGUMENT-OF(left), ARGUMENT-OF(parent))
    else if comb = (>) then
        {X/Y   Y → X}
        COPY-MODES(RESULT-OF(left), parent)
    else if comb = (<) then
        {Y   X/Y → X}
        COPY-MODES(RESULT-OF(right), parent)
    end if
end for
```

```
APPLY-SUBSTITUTION(C, substs):
for each derivation D in corpus B do
    for each category C in a leaf of D do
        if substs has a substitution C′ for category C then
            substitute C′ for C
        end if
    end for
    PERCOLATE(D)
end for
```

Figure 4.19: Full mode annotation and mode percolation algorithm

4.5 Mode splitting: trading categorial ambiguity for efficiency

The MMCCG corpus obtained through the efforts of Section 4.4 serves as the first version of our iteratively improved corpus. In this section, we discuss changes to our approach with the goal of developing a corpus which better utilises the flexibility and power of multi-modal CCG.

At this stage, our corpus annotation procedure does not allow us to partition the set of lexical items assigned a given CCG category into subsets with distinct MMCCG categories. Recall that in Section 4.3.3, we distinguished two classes of adverbs, sentential and VP adverbs, which share a CCG category $(S\backslash NP)/(S\backslash NP)$. We showed that this category is insufficient to express the freedom of movement which characterises VP adverbs, to the exclusion of sentential adverbs, and proposed a MMCCG solution which assigns sentential adverbs the MMCCG category $(S\backslash NP)/_{\star}(S\backslash NP)$, while VP adverbs receive the more permissive MMCCG category $(S\backslash NP)/_{\circ}(S\backslash NP)$.

Ideally, the corpus annotation framework should allow for the splitting of a CCG structural category into multiple MMCCG categories. Once this is possible, a supertagger can, for example, choose to assign a MMCCG category allowing composition precisely when contextual information suggests that it is likely to be required, and assign a more restrictive category otherwise. If the C&C parser fails to find a spanning analysis given the supertagger's current assignment of categories, it has the ability to request a new category assignment from the supertagger, in an attempt to find a spanning analysis. If the parser is unable to find a spanning analysis because the modes on the assigned category are too restrictive, we should ensure that it is still able to fall back on a less restrictive category in order to minimise lost coverage.

As an example, consider a supertagger assigning a category to the lexical item *the* for the two derivations in Figure 4.17. With the appropriate contextual predicates, a supertagger could make the assignment of $NP[nb]/_{\circ}N$ in the case of Example 4.17b. A supertagger could associate the presence of the conjunction *and* in the neighbourhood of the token *the* with the possibility of an analysis such as Example 4.17b, and assign the more permissive mode, while assigning the less permissive $NP[nb]/_{\star}N$ when the context does not suggest involvement in a coordination construction.

How much coverage do we lose with the single mode assignment approach of Section 4.4? In other words, how many derivations no longer yield a spanning analysis because our assignment of modes is too restrictive? Anticipating the parser modifications of Chapter 5, Table 4.5 shows the number of parse failures on the unmoded CCG corpus compared to the number of parse failures on the moded corpus of Section 4.4, as reported by the C&C parser. C&C imposes a frequency cutoff on categories: if a category occurs less than 10 times in the corpus, the parser will not recognise it, leading to *Missing category* errors. *Missing unary* and *missing binary rule* errors result from attempted

combinatory rule applications which are blocked by the parser.

We also compute the number of derivations in the single-mode corpus which fail because a mode assignment is so restrictive that it blocks a derivation previously allowed in the original corpus. This figure allows us to consider the loss of coverage incurred by the addition of modes. In addition to the failure counts returned by the C&C parser, we also independently compute statistics on the corpus to determine the number of failures which overrestrictive modes account for. Note that the combined number of unary and binary rule failures reported by C&C on Sections 02-21 of 1151 errors is smaller than the 2007 mode failures reported by our own independent validation. This deficit is explained by the fact that there are a great deal of noisy rules which C&C, but not our mode-error reporting script accounts for. The true number of mode failures is likely to be less, although due to time constraints, our reporting script does not presently recognise the full set of rules accepted by C&C.

Parse failures in C&C against original corpus as reported by parser

	Failure					
Section	Missing category		Missing unary		Missing binary	
00	54/1913	(2.82%)	2/1913	(0.10%)	21/1913	(1.10%)
02-21	1276/39604	(3.22%)	98/39604	(0.25%)	461/39604	(1.16%)
23	73/2407	(3.03%)	7/2407	(0.29%)	21/2407	(0.87%)

Parse failures in C&C against single-mode corpus as reported by parser

	Failure					
Section	Missing category		Missing unary		Missing binary	
00	54/1913	(2.82%)	1/1913	(0.05%)	51/1913	(2.67%)
02-21	1276/39604	(3.22%)	91/39604	(0.23%)	1060/39604	(2.68%)
23	73/2407	(3.03%)	6/2407	(0.25%)	44/2407	(1.83%)

Mode failures in single-mode corpus as computed independently

Section	*Failure*	
00	97/1913	(5.07%)
02-21	2007/39604	(5.06%)
23	146/2407	(6.07%)

Table 4.5: Parse failures in C&C against unmoded and single-mode corpora

As we would expect, the number of parses which failed due to a "missing binary rule" (the attempted use of a combinatory rule disallowed by the parser due to the constraints on that rule) has greatly increased in the new corpus. This is precisely due to over-restrictive modes: a derivation which originally succeeded now fails because the parser, heeding the mode annotations, no longer considers a combinatory rule which is

not licensed by the modes on its input categories. Although the loss of coverage does not stop the parser from acquiring a model, the greater a quantity of training data is available, the better the model.

To reduce the coverage lost by assigning overly restrictive modes, and to enable the supertagger to make more efficient category assignments, we identify slashes whose distribution suggests that we could split the single category assigned through the process of Section 4.4 into multiple categories distinguished by modes. Currently, we select the candidates for mode splitting manually, based on linguistic domain knowledge (such as the differentiation of the sentential and VP adverbs of Section 4.3.4) and by examining the slash/combinator analysis of Table 4.2.

Although splitting a structural CCG category such as $(S\backslash NP)\backslash(S\backslash NP)$ into multiple MMCCG categories such as $(S\backslash NP)\backslash_{\star}(S\backslash NP)$ and $(S\backslash NP)\backslash_{\circ}(S\backslash NP)$ certainly increases the amount of lexical ambiguity, by making this distinction we hope to better utilise the supertagger, which is responsible for making category assignment distinctions, and in turn determine whether the increase in categorial ambiguity also increases the parser's efficiency.

4.5.1 Approach to mode splitting

We describe a number of experiments to comparing supertagger accuracy (the proportion of correct category assignments), as obtained with several versions of the original corpus. The rationale for mode splitting is to make the category assignment task more difficult for the supertagger (by increasing categorial ambiguity) with the goal of rendering the parser more efficient.

The goal of the experiments of this section is to determine *how* much more difficult we have made the job of the supertagger, and whether it is feasible to trade categorial ambiguity in the supertagger for efficiency and precision in the parser. We will consider multiple experiments, differing by the set of categories on which mode splits are performed. Since the work of this section examines the impact of mode splits on the supertagger separately to the parser, we use the standard metric of supertagger accuracy (SUP, as described in Section 3.2.1), considering its impact on the parser later.

4.5.2 Experimental framework for mode splitting

We consider three sets of experiments, the first of which is the baseline of the unmodified CCGbank, corresponding to C&C prior to any of our modifications. Next, we consider a corpus which maps only one MMCCG category to each structural category, corresponding to the corpus derived in Section 4.4.

In the supertagging phase, before the parser is even invoked, categories are simply opaque labels in a classification problem: the goal is simply to map one label (a category) to each item of input (a word). Accordingly, a supertagger should consider

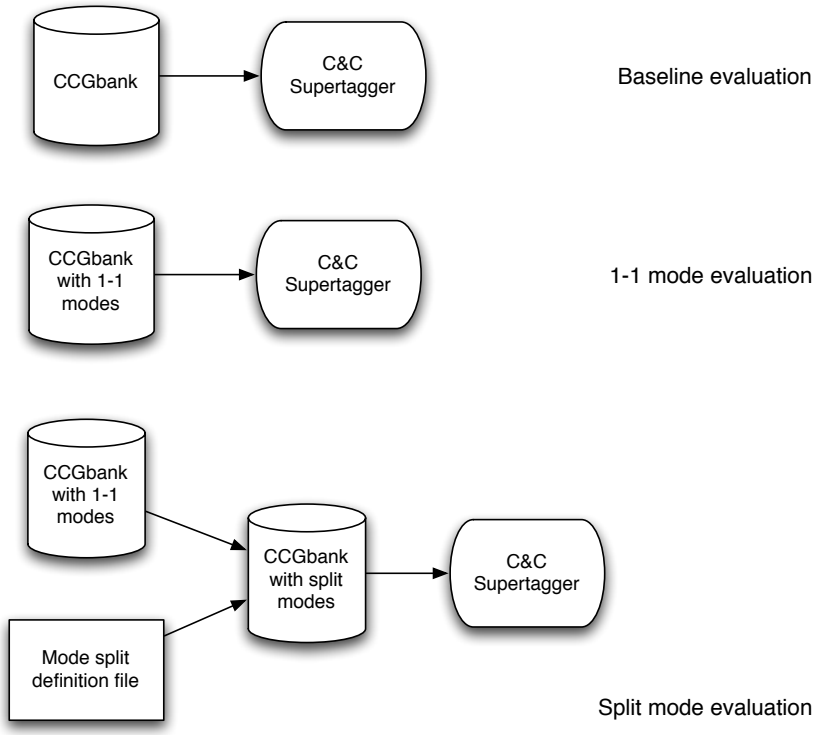


Figure 4.20: Experimental setup for mode splitting

the unmoded CCGbank and a one-to-one mode-mapped corpus functionally identical, because we have simply substituted exactly one moded category for every unmoded category. Accordingly, although we expect the supertagger results from the unmoded corpus to differ insignificantly from those of the one-to-one mode-mapped corpus, we reproduce the figures here as a sanity check, and also since we will consider the effect of each of these sets of corpora on the parsing task, in which we expect that each of the three sets of corpora will behave differently.

The third set of corpora forms the core of our evaluation: the mode-split corpora. The generation of a mode-split corpus is controlled by a split definition file, which specifies which structural categories to split, and the allowed MMCCG categories with which to replace the structural categories.

```
NP[nb]/N 0
%
NP[nb]/N NP[nb]/*N
NP[nb]/N NP[nb]/@N
```

Figure 4.21: Simple split definition file

```
(S[dcl]\NP)/NP 0
%
(S[dcl]\NP)/NP (S[dcl]\*NP)/*NP
(S[dcl]\NP)/NP (S[dcl]\*NP)/@NP
(S[dcl]\NP)/NP (S[dcl]\@NP)/@NP
```

Figure 4.22: Additional constraints in a split definition file

The split definition file in Figure 4.21 specifies that every leaf with structural category $NP[nb]/N$ in the input corpus is to be examined, and that the two possible MMCCG categories it may take are $NP[nb]/_{\star}N$ and $NP[nb]/_{\circ}N$. In the output corpus, every leaf with structural category $NP[nb]/N$ will be assigned one of these alternatives according to whether the slash in question was consumed by application, or a non-application combinatory rule.

For full generality, our system allows for a split definition file such as Figure 4.22. In this example, the candidate slash for splitting is slash 0 of $(S[dcl]\backslash NP)/NP$, and we are constrained to replace it only with one of the three provided alternatives. Slash 0 will either receive the mode $\star$, or the mode $\circ$. However, there are two possible categories with $\circ$ in slash 0, differing in the permissibility of slash 1 of each alternative. As much as possible, we do not want to assign a category which is any *more* permissive than the original category, since the less permissive a slash is, the smaller the set of combinatory rules the parser can attempt to apply.

Following these constraints, Figure 4.23 gives an algorithm for mode splitting, which assigns the *least permissive category which preserves the original analysis*. First, we explain some conventions from the algorithm. The function ZIP takes two lists A, B and returns a list of tuples $\langle (A_1, B_1), (A_2, B_2), \dots \rangle$. Also, we define the mode vector in this way:

DEFINITION 5. *The mode vector of an atomic category is the empty list. The mode vector of a compound category $X|_iY$ is the mode i, followed by the mode vector of X, followed by the mode vector of Y.*

For example, the mode vector of $(S\backslash_{\star}NP)\backslash_{\circ}(S\backslash_{\bowtie}NP)$ is $\langle\circ,\star,\bowtie\rangle$. Lastly, the notation PERMISSIVENESS$(C,\cdot,j)$ in Line 13 of the algorithm means that we seek the element $a\in A$ which minimises the value PERMISSIVENESS(C,a,j).

We will explain the algorithm through an example. Consider the split definition given in Figure 4.22. Suppose our mode-splitting algorithm encounters $(S[dcl]\backslash_{\star}NP)/_{\star}NP$ as a leaf category. This matches the structural category $(S[dcl]\backslash NP)/NP$, so we consider slash 0 of the category to be a candidate for mode splitting. Suppose that we determine the combinatory rule that consumed slash 0 of $(S[dcl]\backslash_{\star}NP)/_{\star}NP$, and find it to have been consumed by composition. To preserve the analysis, we only consider the alternatives whose slash 0 is compatible with consumption: $(S[dcl]\backslash_{\star}NP)/_{\circ}NP$ or $(S[dcl]\backslash_{\circ}NP)/_{\circ}NP$). Out of these two, our choice is the category which is the least permissive relative to the original category $(S[dcl]\backslash_{\star}NP)/_{\star}NP$, ignoring the candidate slash. The category which satisfies this condition is $(S[dcl]\backslash_{\star}NP)/_{\circ}NP$, so we select it to replace the original category $(S[dcl]\backslash_{\star}NP)/_{\star}NP$.

The algorithm we give has the effect of making slashes which are *too* permissive less permissive, and slashes which are *too* restrictive less restrictive. The first property improves efficiency, while the second property improves coverage.

4.5.3 Specifying the targets of mode splitting

Having established our framework for creating corpora which embody different mode splits, how do we decide which structural categories to split? Again, we have two extremes: automatic or manual annotation, with degrees of automation in between. In developing an approach, we consider what attributes of the distribution of a category suggest that mode-splitting would be useful.

Frequency of consumption by application or non-application rules: We are interested in those slashes whose distribution does not strongly suggest that they should be made application-only ($\star$), but rather have a mixed distribution, being consumed sufficiently frequently by both application and non-application combinatory rules. We wish to harness the supertagger's power to distinguish, based on local context, each of the alternatives in this mixed distribution.

Contextual predicates discriminate between category alternatives: When tagging a word, the supertagger has access to a selected local context of that word, in the form of contextual predicates. To exploit this local information in the supertagger to distinguish between alternatives, there must be some aspect of the context which has the potential to distinguish between the alternatives. For example, the local presence of a conjunction such as *and* may allow a supertagger to prefer a category with a first slash permitting composition, in order to allow an analysis similar to those of Section 4.3.4.1.

TIER($mode$):

if $mode = \bowtie$ **then**
  return 0
else if $mode = \star$ **then**
  return 1
else if $mode = \circ$ **then**
  return 2
else
  {Mode must be in $\{\bowtie, \star, \circ\}$}
end if

PERMISSIVENESS(P, Q, j):

$result \leftarrow 0$
$m^P, m^Q \leftarrow$ mode vectors for P and Q
for each (m_i^P, m_i^Q) in ZIP(m^P, m^Q), where $i \neq j$ **do**
  if TIER(m_i^P) > TIER(m_i^Q) **then**
    $result \leftarrow result + 1$
  end if
end for
return $result$

DO-MODE-SPLITS(B):

for each derivation D in corpus B **do**
  for each category C in a leaf of D **do**
    for each slash j in category C **do**
      if slash j has mode $\star$ and is actually consumed by composition **then**
        FIX-MODE($C, j, \circ$)
      else if slash j has mode $\circ$ and is actually consumed by application **then**
        FIX-MODE($C, j, \star$)
      end if
    end for
  end for
end for

FIX-MODE(C, j, m)

$\mathcal{C} \leftarrow$ set of alternatives which match the structural category of C
$A \leftarrow \{c \in \mathcal{C} \mid$ slash j of c has mode $m\}$
return if A is empty
if slash j of category C is a candidate for mode splitting **then**
  $m^C \leftarrow$ mode vector of C
  for each alternative category $a \in A$ **do**
    $m^A \leftarrow$ mode vector of A
    if for any $i \in \{0, 1, \ldots, |m^A|\}$, mode m_i^A is less permissive than mode m_i^C **then**
      Remove a from the set of alternatives A
    end if
  end for
  {If A is empty, fail: there are no valid alternatives}
  Replace C with MINIMUM-BY(PERMISSIVENESS($C, \cdot, j$), A)
end if

Figure 4.23: Algorithm for mode splitting

We can directly and automatically determine the suitability of a slash for mode splitting with respect to the first criterion, by examining a slash/combinator analysis (as in Table 4.2) and choosing those slashes whose frequency of consumption by application rules lie below the application-only threshold α, the boundary above which our automatic mode assignment algorithm of Figure 4.18 would assign the application-only mode $\star$. However, Table 4.6 shows that there is typically an amount of noise, in the form of slashes consumed by composition rules in faulty analyses.

Category	*Slash index*	*Comp.*	*Appl.*
$(S[b]\backslash NP)/(S[pt]\backslash NP)$	0	26	380
$(S[b]\backslash NP)/S[em]$	0	26	216
$(NP/NP)/(S[asup]\backslash NP)$	1	26	3
$(S[pt]\backslash NP)/S[em]$	0	24	118
$(NP\backslash NP)/(NP\backslash NP)$	0	22	1571
$(S[b]\backslash NP)/S[qem]$	0	22	204
$((S\backslash NP)\backslash(S\backslash NP))/S[em]$	0	7	28
$(S/S)/PP$	0	6	339
$(S[pt]\backslash NP)/(S[adj]\backslash NP)$	0	6	322
$(S\backslash S)/S[dcl]$	1	6	280
$((S[b]\backslash NP)/(S[adj]\backslash NP))/NP$	0	6	192

Table 4.6: Selected frequency of consumption by composition vs. application

In an attempt to avoid splitting categories whose mixed distribution with respect to consuming rules is due to noise, we impose a fixed cutoff frequency β: a slash must be consumed at least β times by composition to be considered as a candidate for mode splitting. Accordingly, the criterion a slash/combinator pair must meet to be considered as a candidate for mode splitting is:

$$\beta \leq C_\circ < \alpha \cdot C_*$$

where $C_\circ$ is the frequency of consumption by composition, and C_* is the total frequency with which the slash occurs, and α is the application/composition-only threshold from the algorithm of Figure 4.18.

4.5.4 Candidate slashes for mode splitting

The sets of candidates for mode splitting, as used in the experiments of the following sections, is given in Table 4.7. We create corpora which only split on one category (such as $NP[nb]/N$) to allow us to determine the extent to which the supertagger successfully uses contextual information to choose from a set of moded alternatives (such as $NP[nb]/_\star N$ and $NP[nb]/_\circ N$). We also evaluate a set of split candidates obtained automatically through the procedure described in Section 4.5.3.

Corpus name	*Split category*	*Slash*
$NP[nb]/N$	$NP[nb]/N$	0
$(S\backslash NP)\backslash(S\backslash NP)$	$(S\backslash NP)\backslash(S\backslash NP)$	0
N/N	N/N	0
$NP[nb]/N$ and N/N	$NP[nb]/N$	0
	N/N	0
Automatic splits ($\alpha = 0.95, \beta = 500$)	$(S\backslash NP)\backslash(S\backslash NP)$	0
	$(S[dcl]\backslash NP)/(S[b]\backslash NP)$	0
	$(S[dcl]\backslash NP)/NP$	0
	$((S\backslash NP)\backslash(S\backslash NP))/NP$	1
	$(S[b]\backslash NP)/NP$	0
	$(S[dcl]\backslash NP)/(S[pt]\backslash NP)$	0
	$(S[dcl]\backslash NP)/S[dcl]$	0
	$(S[dcl]\backslash NP)/(S[adj]\backslash NP)$	0
	$(S[dcl]\backslash NP)/(S[pss]\backslash NP)$	0
	$(S[dcl]\backslash NP)/(S[ng]\backslash NP)$	0
Manual splits	N/N	0
	$NP[nb]/N$	0
	$(S[b]\backslash NP)/(S[adj]\backslash NP)$	0
	$((S[b]\backslash NP)/NP)/NP$	0
	$(S[b]\backslash NP)/NP$	0
	$(S[b]\backslash NP)/(S[pss]\backslash NP)$	0
	$(S[adj]\backslash NP)/NP$	0
	$(S[pss]\backslash NP)/NP$	0
	$(S[pt]\backslash NP)/NP$	0
	$(S[ng]\backslash NP)/NP$	0
	$(NP\backslash NP)/N$	0
	$(S\backslash NP)\backslash(S\backslash NP)$	0
	$(S\backslash NP)/(S\backslash NP)$	0
	$(S[b]\backslash NP)/(S[adj]\backslash NP)$	0
	$(S[b]\backslash NP)/(S[pss]\backslash NP)$	0

Table 4.7: Candidate slashes for mode splitting

4.5.5 Evaluation of mode-split corpora

Having trained a supertagger on a number of corpora generated with different mode splits, we justify and describe our choice of statistical significance tests to determine whether mode splitting has a significant effect on supertagger performance. Gillick and Cox [1989] presents three statistical tests which may be used in evaluating the performance of two algorithms A_1 and A_2 on a labelling task – supertagging is simply a labelling task which assigns category labels to input words.

The first of three tests is a variation on the t-test, where the null hypothesis is that the difference in the error rates p_1 and p_2 of algorithms A_1 and A_2 respectively is zero. The

key assumption of this test is that the error rates of the two algorithms are independent, a condition which is not satisfied when the two algorithms are evaluated on the same dataset. We can see this by considering that algorithms which make similar choices will fail in similar ways on the same input.

The second test is *McNemar's test*, where the null hypothesis is that given that exactly one algorithm A_i made an error, it is equally as likely that either A_1 or A_2 made the error. McNemar's test models the distribution of errors made by only one of the two algorithms as binomial, such that the test assumes that a tagging error on one token is independent of previous tagging errors. This assumption is satisfied when the error in tagging a single token is independent of the context, which is clearly untrue for the supertagging problem – Curran et al. [2006] documents that the contextual predicates used in the C&C supertagger include the lexical items and POS tags in a two-word window anchored on the token being supertagged.

The third, the *matched-pairs test* attempts to minimise the impact of context by partitioning a labelling task into *segments*, such that the distribution of error rates between segments is assumed to be independent. This condition is satisfied by the supertagging task: we can treat each sentence as a segment, since the distribution of errors between different sentences is seen to be independent. Given that we ultimately select the matched-pairs test for our evaluation, we characterise it with particular reference to the supertagging problem. Assume that we have k derivations, and two supertagger models A_1 and A_2. We wish to determine whether there is a significant difference between the supertagger accuracy of A_1, and that of A_2.

Comparing the sequence of category assignments performed with model A_i to a set of gold standard category assignments, compute an error vector N_i of length k for each A_i, where each N_i^j is the number of categories in derivation j incorrectly supertagged by model A_i, relative to the gold standard. Now, let the random variable:

$$Z^j = N_1^j - N_2^j$$

be the difference in errors on derivation j between models A_1 and A_2. The null hypothesis $\mathbf{H}_0$ is that the mean difference in errors is zero, while the alternative hypothesis $\mathbf{H}_1$ is that there is a difference in errors.

$$\mathbf{H}_0 : \mu_Z = 0 \qquad \mathbf{H}_1 : \mu_Z \neq 0$$

Having computed the sample mean $\hat{\mu}_Z$ and variance $\hat{\sigma}_Z^2$, we define the test statistic W. We can see that the form of the test statistic W is identical to that of the standard t-test for equality between the means of two samples [Rice, 1995], except the mean of the

second sample is taken to be zero, in line with the null hypothesis above.

$$W = \frac{\hat{\mu}_Z}{\sqrt{\hat{\sigma}_Z^2/n}}$$

W is distributed as $Z \sim N(0,1)$ as k becomes large. We compute a two-tailed p-value below, corresponding to the alternative hypothesis that the mean difference is not zero.

$$p = 2P(Z \geq w)$$

If the p-value is less than a significance level α, then the matched-pairs test indicates a significant difference in supertagging accuracy between the two models A_1 and A_2 at significance α.

We can see that the matched-pairs analysis attempts to minimise the impact of violating the independence constraint by assuming that the distribution of errors between different sentences is independent.

4.5.6 Analysis of results

We present the results of the matched-pairs analysis on the seven corpora of the previous section.

Corpus	*Accuracy and p-value*	
Baseline (unmodified CCGbank)	92.57%	–
One-to-one mode mapped	92.49%	0.21
$NP[nb]/N$	92.40%	0.007
$(S\backslash NP)\backslash(S\backslash NP)$	92.38%	0.003
N/N	92.30%	9.2×10^{-5}
Automatic splits	92.27%	2.12×10^{-5}
$NP[nb]/N$ and N/N	92.26%	6.18×10^{-6}
Manual splits	92.18%	1.52×10^{-7}

Table 4.8: Analysis of change in supertagger accuracy relative to the baseline

Table 4.8 shows that as expected, the difference between the baseline and one-to-one mode mapped corpus is not significant at level $\alpha = 0.05$. The fact that the two figures differ at all is because of the tie-breaking algorithm in the supertagger, which depends on the lexicographic order of a set of tied candidates. Although it is true that one-to-one mode mapping simply maps from one structural category to one MMCCG category, the newly added mode symbols in the MMCCG categories affect their sorting order. We further confirmed this fact by stripping out all occurrences of modes from the one-to-one mode mapped corpus, and ensuring that the resulting evaluation figures were identical to the baseline.

We can see that any mode splitting reduces supertagger accuracy in a statistically significant manner. We hypothesise that the window size of two tokens on either side of

the word under consideration is too small to capture useful contextual cues that reliably distinguish between category alternatives. Although it lies outside the scope of this present work, as it would require modifications to the supertagger component, in future work we intend to validate our hypothesis for the reduction in supertagger accuracy by performing the same analysis as the above, with larger sizes of supertagger window. An increase in supertagger accuracy following an enlargement of the supertagger window would suggest that indeed, a 2-token window is insufficient context to discriminate between fine-grained alternatives.

4.6 Complete annotation framework

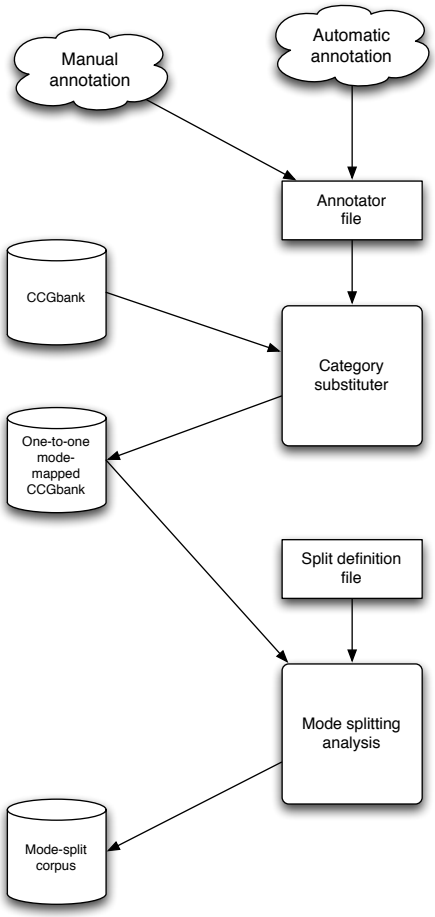


Figure 4.24: Corpus generation framework

We conclude by describing the structure of the system used to produce the one-to-one mode-mapped corpus and the various mode-split corpora. The generation of the one-to-one mode-mapped corpus is driven by an annotator file, which aggregates the results of automatic annotation from the work of Section 4.4.2, as well as the slashes manually annotated according to the analysis of Section 4.1. Each line of the annotator file maps one CCG category to one MMCCG category. In generating the simple one-to-one mode-mapped corpus, we simply use this annotator file as a substitution map to obtain a corpus which consistently replaces every CCG category with one MMCCG category.

The refinements of Section 4.5 increase the number of MMCCG categories to which a CCG category may be mapped, to better exploit the context-extracting power of the supertagger while increasing the efficiency of the parser. We generate these corpora by a separate process, using the one-to-one mode-mapped corpus as input. As described in Section 4.5.1, this process is driven by a split definition file, which identifies candidate categories for mode splitting, and defines the possible alternatives to which a CCG category may be mapped. Depending on the split definition file used, we achieve the various mode-split corpora on which we performed the evaluation of Section 4.5.6.

4.7 Conclusion

Through the analysis described in this chapter, we have reformulated the mode hierarchy of Baldridge [2002] for our purposes, justifying the presence of our mode system through linguistic analysis of the generative power required to analyse the syntax of English. Although the analysis we have done is specific to English syntax, we believe that the general approach we have employed: manual linguistic analysis combined with automatic annotation, can be applied to arbitrary languages.

We have successfully produced various moded corpora which we will use to assess the impact of trading various degrees of categorial ambiguity for efficiency in the parser. We have described two models of corpus derivation: one which maps each CCG category to one MMCCG category, and one which further refines the category assignment to map each CCG category to a set of MMCCG categories according to need. In the following chapter, having obtained a practical parser with which we can evaluate the corpora generated by the work described in this chapter, we will examine our claims of increased parser efficiency by creating, and training a MMCCG parser on the various corpora we have obtained.

CHAPTER 5

Augmenting a CCG parser for MMCCG

The third contribution of this work is to make the modifications necessary to the C&C parser to enable it to interpret, process and make use of the multi-modal CCG corpus we derived through the procedure described in Chapter 4. This chapter describes the structural changes necessary to augment the parser for multi-modal CCG, culminating with an evaluation of the completed MMCCG parser on our new corpus, relative to the unmodified C&C parser on the original CCGbank. Before describing the changes we have made to the C&C parser, we describe its general structure and give an overview of the implementation, and the design decisions which it incorporates.

5.1 Introducing C&C

C&C consists of 49 000 lines of highly optimised C++, implementing the key components of a statistical parser: a supertagger and parser, together with algorithms for training the two, a statistical model of derivation likelihood for parse selection, and a front-end which allows for its embedding in natural language applications.

The system is engineered for efficiency, and as such, its design reflects "handling common cases efficiently" over strict adherence to the formalism, and the sacrifice of generality and extensibility for reductions in memory and time requirements. An example of a concession to efficiency made in C&C at the cost of generality is its implementation of only those combinatory rules which are necessary to analyse English as represented in CCGbank. For example, forward crossed composition is entirely unimplemented in C&C, due to its effect in English as discussed in Section 4.3.4. Accordingly, to implement an augmentation of the base formalism, such as MMCCG, structural modifications are necessary in several parts of the parser. The changes of this section centre on the parser component, which must be modified to accept and respect the mode annotations of MMCCG.

The ensemble structure of C&C is demonstrated by Figure 5.1, which shows the sources of input data required to train and evaluate the system. We describe the typical flow of data through C&C during the training and evaluation process. First, C&C extracts gold standard dependencies from the training data, converting them to the dependency representation used by the system. Then, the C&C parser is invoked on the corpus, to

filter out the gold-standard derivations that the parser cannot actually produce. As discussed in Section 4, noise from various sources in CCGbank results in derivations whose derivation structure implies the use of unorthodox combinatory rules. Since these derivations cannot be parsed, let alone be used as training data for the system, they are removed from consideration at this point. To maximise the amount of training data available to C&C, a more permissive set of combinatory rules is activated in the training data acquisition phase. This allows derivations which are still of value as training data, but would otherwise be rejected if parsed through the normal set of combinatory rules, to be allowed strictly as training data.

Next, the supertagger is trained on the set of valid derivations, to acquire a model of the distribution of category assignments with respect to the context of each lexical item. Subsequently the C&C parser is trained on the data, with a numerical optimisation algorithm known as L-BFGS (Limited-memory BFGS) used to train the parser model [Clark and Curran, 2007]. In C&C, this numerical optimisation step is distributed and parallelised to run over a cluster of machines.

At this point, we have a complete parser model ready for evaluation. For each derivation in the test set, the supertagger assigns a set of one or more categories to each lexical item, and the parser produces a packed chart, a data structure which encodes the set of possible parses over that derivation. Finally, a statistical model chooses the most likely parse, and the dependencies for the selected parse are submitted as the parser's output for that derivation. The dependency-based evaluation of Section 3.2 is conducted on the parser output to yield the previously discussed set of evaluation metrics.

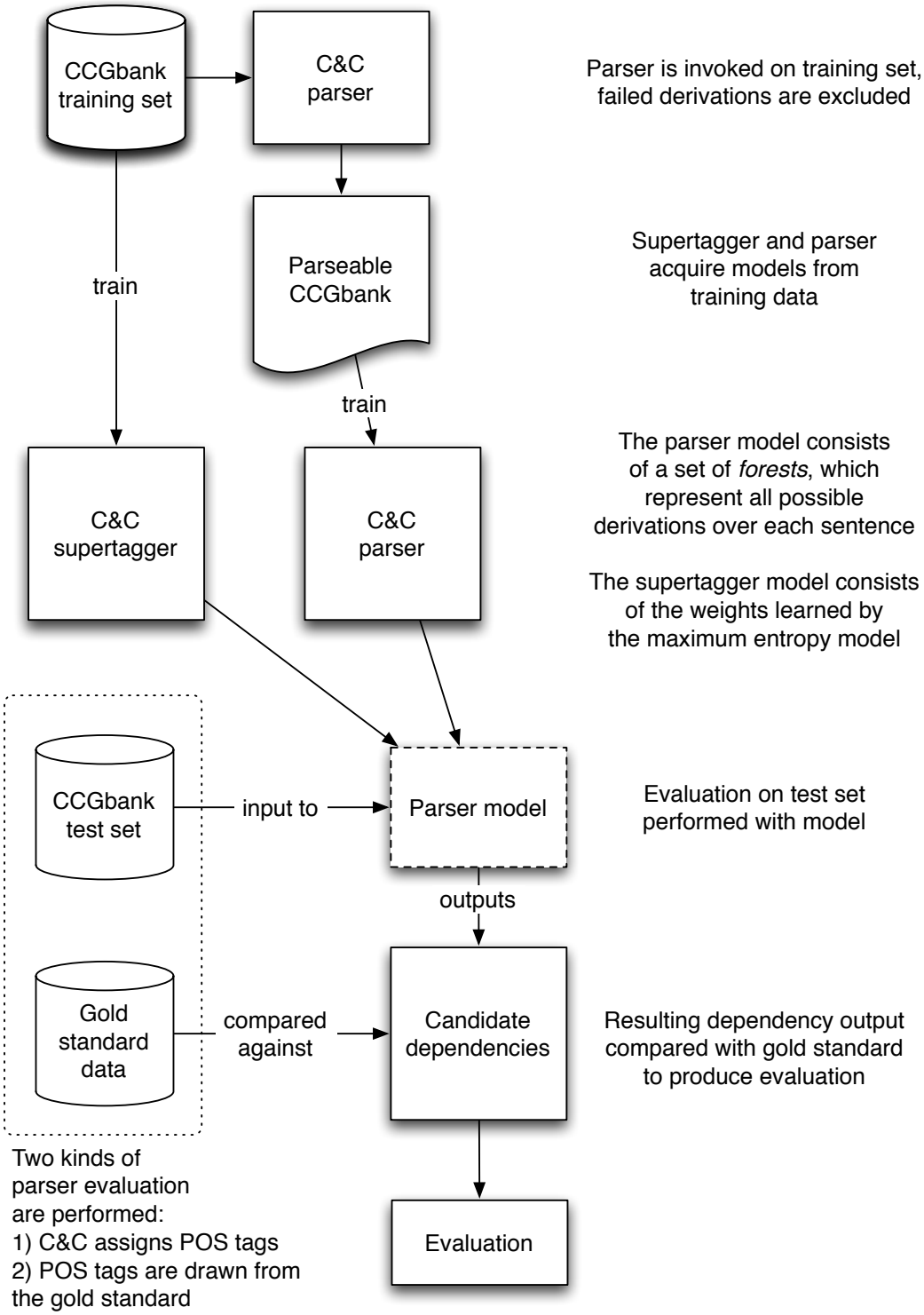


Figure 5.1: c&c training and evaluation process

5.2 Modifying the representation of categories

The key contribution of MMCCG is the annotation of each slash with a mode, which limits the set of combinatory rules by which that slash can be consumed. In C&C, in order to avoid the overhead of virtual method calls in C++ incurred by inheritance, both atomic and compound categories are represented with the same class, `Cat`. The data of an atomic category are contained in a field `atom`, and a compound category is simply a `Cat` whose `atom` field is `NULL`.

Furthermore, a degree of caching is performed: a bitfield `flags` answers various extremely common queries on categories. This bitfield allows these queries to be returned directly without examining the category structure itself. The intuition behind `flags` is that categories are created, queried and combined together with very high frequency in the course of parsing. In particular, common queries on categories, to the extent possible, should be extremely fast. This caching allows these checks to be performed without performing the recursive comparison of category structure which would otherwise be required. A disadvantage is that the cached attributes reflect queries which are common in parsing English as encoded by CCGbank: a change of corpus, or language will surely call upon a different set of attributes. This tradeoff of generality for efficiency is another example of the C&C implementation philosophy.

Instead of representing modes in a separate field, we employ the same bitpacked representation to store them within `flags`. This allows us to check the mode of a slash together with other structural conditions in a single operation. In adding new modes to the mode hierarchy, we are limited by the number of bits not yet assigned a significance in `flags`. Also, with the present approach, the behaviour of modes, and the implicit mode inheritance hierarchy, is hard-coded in parser behaviour, due to the C&C representation of modes as simple flags. By comparison, representing modes as objects simplifies the process of modifying the mode hierarchy, while the mode implementations themselves can specify their behaviour. In line with the implementation philosophy of C&C, we prefer to represent modes in a manner consistent with existing category attributes such as slash directionality to allow efficient checking for modes, at some cost to generality. The set of flags after our modifications is given in Figure 5.2.

Bit index	*Interpretation*
0	Slash direction
1	Category result is *S*
2	Category argument is *S*
3	Category result is *NP*
4	Category argument is *NP*
5	Category result is *N*
6	Category argument is *N*
7	Category result is *S**NP*
8	Application-only mode ($\star$)
9	Null mode ($\bowtie$)
10	Maximally permissive mode ($\circ$)

Figure 5.2: Flags defined on categories in C&C

$$
\begin{array}{rcl}
\langle \text{category} \rangle & \longrightarrow & \langle \text{complex} \rangle \mid \langle \text{basic} \rangle \\
\langle \text{complex} \rangle & \longrightarrow & \texttt{(}\ \langle \text{category} \rangle\ \langle \text{slash} \rangle\ \langle \text{category} \rangle\ \texttt{)} \\
\langle \text{slash} \rangle & \longrightarrow & \texttt{/} \mid \texttt{\textbackslash} \\
\langle \text{basic} \rangle & \longrightarrow & \langle \text{atom} \rangle\ [\langle \text{feature} \rangle]\ [\langle \text{variable} \rangle]\ [\langle \text{slot} \rangle] \\
\langle \text{atom} \rangle & \longrightarrow & (\text{alpha} \mid \texttt{,} \mid \texttt{.} \mid \texttt{;} \mid \texttt{;})^{+} \\
\langle \text{feature} \rangle & \longrightarrow & \texttt{[}\ \text{alpha}^{+}\texttt{]} \\
\langle \text{variable} \rangle & \longrightarrow & \texttt{\{}\ (\text{alpha} \mid \texttt{\_} \mid \texttt{*})^{+}\texttt{\}} \\
\langle \text{slot} \rangle & \longrightarrow & \texttt{<}\ \text{num}^{+}\texttt{>}
\end{array}
$$

Figure 5.3: Grammar recognising CCG category representations

5.3 Modifying input handling

C&C implements a hand-coded recursive descent parser on a small grammar to parse the representation of categories. The grammar defining the structure of a C&C category is shown in Figure 5.3.

For each category string, the category parser yields an instance of `Cat`, the C&C internal representation of a category. Our modifications to `Cat` add bits representing the modes on every compound category, which we need to populate from the string representation of a MMCCG category. Our convention is to represent each mode in the various MMCCG corpora as an ASCII character following the slash, similar to the notation used by Baldridge in Grok to indicate modes.

With these conventions, we can supplement the grammar for the C&C category parser to receive the additional mode symbols. Although the grammar stipulates that modes are obligatorily indicated in the string representation of a category, we allow the category

ASCII	*Mode symbol and description*
-	$\bowtie$ Null mode
*	$\star$ Application-only mode
@	$\circ$ Maximally permissive mode

Figure 5.4: Mapping of modes to ASCII representations

$$
\begin{array}{rcl}
\langle \text{category} \rangle & \longrightarrow & \langle \text{complex} \rangle \mid \langle \text{basic} \rangle \\
\langle \text{complex} \rangle & \longrightarrow & \texttt{(}\ \langle \text{category} \rangle\ \langle \text{slash} \rangle\ \langle \text{mode} \rangle\ \langle \text{category} \rangle\ \texttt{)} \\
\langle \text{slash} \rangle & \longrightarrow & \texttt{/} \mid \texttt{\textbackslash} \\
\langle \text{mode} \rangle & \longrightarrow & \texttt{@} \mid \texttt{*} \mid \texttt{-} \\
\langle \text{basic} \rangle & \longrightarrow & \langle \text{atom} \rangle\ [\langle \text{feature} \rangle]\ [\langle \text{variable} \rangle]\ [\langle \text{slot} \rangle] \\
\langle \text{atom} \rangle & \longrightarrow & (\text{alpha} \mid \texttt{,} \mid \texttt{.} \mid \texttt{;} \mid \texttt{;})^{+} \\
\langle \text{feature} \rangle & \longrightarrow & \texttt{[}\ \text{alpha}^{+}\texttt{]} \\
\langle \text{variable} \rangle & \longrightarrow & \texttt{\{}\ (\text{alpha} \mid \texttt{\_} \mid \texttt{*})^{+}\texttt{\}} \\
\langle \text{slot} \rangle & \longrightarrow & \texttt{<}\ \text{num}^{+}\texttt{>}
\end{array}
$$

Figure 5.5: Grammar recognising MMCCG category representations

parser to interpret the absence of any of the mode symbols in Figure 5.4 as equivalent to specifying the maximally permissive mode ($\circ$).

At this point, we have a version of C&C which reads in and interprets modes, albeit without assigning them any significance in the parser. However, we are able to verify that our modifications were successful, by reading in the modified CCGbank of Chapter 4, and observing that the parser category output correctly renders modes.

5.4 Enforcing modality constraints in the parser

The key to the efficiency of MMCCG is the restrictions it allows us to place on the arguments to a combinatory rule. For example, if at least one of the arguments to an attempted composition carries the mode $\star$ or $\bowtie$, then we can abandon the consideration of that rule immediately.

When deciding which combinatory rule to apply, C&C is able to immediately rule out certain rules based on the `flags` of each of the input arguments. For example, forward application (>) is impossible unless the first argument is a compound category and has a forward slash.

Having augmented the C&C representation of categories to accommodate the assignment of modes, we impose constraints on the parser to reject the consideration of combinatory rules which are blocked by an assignment of modes. This is the key

modification which enables the parser to consider fewer combinatory rules, when the slashes of the input categories carry restrictive modes. For each combinatory rule implemented in c&c, there exists a function which examines structural properties of the input categories, and decides whether the parser should consider that rule. We modify these functions to reject a given rule if it is incompatible with the modes on its input categories. This modification has enabled c&c to apply the constraints encoded in mmccg modes. At this point, we have a version of c&c which we can use to evaluate the impact of mmccg constraints on a wide-coverage parser.

5.5 Evaluating the impact of modes on the modified C&C

Throughout this work, we have attributed increased efficiency with respect to time and space as two of the benefits of mmccg over the pure ccg, in abstract terms. The work of this chapter has, for the first time, allowed us to evaluate these claims in a quantitative manner, with the creation of a wide-coverage mmccg parser. In this section, we discuss how exactly we measure these benefits, with particular reference to the implementation of the modified c&c parser we have obtained. To explain our choice of metric, we need to elaborate on the c&c parsing process, and discuss the internal structures which are created as the parser processes a sentence.

cky is a bottom-up dynamic programming parsing algorithm for the context-free languages. For a sentence of n words, it produces a $(n+1)\times n$ tableau, such that cell C_{ji} contains the non-terminals which can span indices j to i of the input. cky is simple to describe: for every end index i for a span, for every valid start index j, for every index k between j and i, the non-terminals spanning (j,i) are those spanning (j,k) together with those spanning (k,i).

cky($\langle s_0,\ldots,s_{n-1}\rangle$):
1: **for each** $1 \le i \le n$ **do**
2: $\quad C_{i-1,i} \leftarrow$ non-terminals producing s_{i-1}
3: **end for**
4: **for each** end index $1 \le i \le n$ **do**
5: $\quad$ **for each** start index $i-2 \ge j \ge 0$ **do**
6: $\quad\quad$ **for each** intervening index $j+1 \le k \le i-1$ **do**
7: $\quad\quad\quad L \leftarrow C_{jk}$
8: $\quad\quad\quad R \leftarrow C_{ki}$
9: $\quad\quad\quad C_{ji} \leftarrow \{$non-terminals producing $L\} \cup \{$non-terminals producing $R\}$
10: $\quad\quad$ **end for**
11: $\quad$ **end for**
12: **end for**
13: **return** $C_{0,n}$

Figure 5.6: cky algorithm for context-free languages

A defining feature of CKY is that the grammar must be in *Chomsky normal form*: that is, each production either has two non-terminals or a single terminal on its right hand side, and only the start symbol can generate the empty string. CKY is easily modified for CCG parsing, as noted by Steedman [2000]: instead of C_{ji} storing the set of non-terminals which span j to i, it stores the set of *categories* with that span instead. It is also easy to allow CKY to handle unary combinatory rules. Recognition algorithms for several mildly context-sensitive grammar formalisms are analysed in Vijay-Shanker and Weir [1993], and a recognition algorithm for CCG is also given in Steedman [2000].

C&C uses a *packed chart representation*, which lets a partial derivation which produces the same category and the same set of unfilled dependencies share a cell in the chart. The intuition is that entries that share a cell can be treated exactly alike for the purposes of the parser. With the packed chart, if a local structure is re-used many times in the same derivation, it is only stored once, rather than multiple times, as it would be in the conventional chart representation, with each occurrence in the derivation of this local structure being replaced by a reference to that single cell [1].

When combining two cells in a packed chart, as may occur during the parsing process, a data structure known as a conjunctive node is formed internally to represent the combined packed chart cells. The number of conjunctive nodes created in the course of parsing a sentence is hence indicative of the size of the packed chart produced during the parsing process. We will use the *number of conjunctive nodes* as our criterion for evaluating the memory requirement characteristics of each corpus. The reason why we do not use a simpler measure, such as the total amount of system memory consumed, is because C&C allocates memory from a pool, and only releases it at the end. Accordingly, when parsing a batch of sentences, memory is allocated once, grown as necessary (for example, when a particularly long sentence is encountered) with the allocated storage being reused between derivations. Total system memory allocated is not a meaningful measure of memory requirement characteristics, because the allocation behaviour of C&C means that this value will only reflect the maximum amount of memory used by some sentence.

The time consumption characteristics of the parser, however, are much more straightforward to determine. The three metrics output in the standard parser log are *total parsing time*, *sentence speed* and *word speed*, reflecting the system time taken to parse the entire set of sentences, and the average parsing rates in sentences or words per second.

[1] Although Eisner normal form (described in Section 2.2.4) eliminates a great deal of spurious ambiguity, in practice, the derivation structure of CCGbank introduces sources of ambiguity which Eisner normal form does not exclude. Accordingly, packed charts are still useful in further reducing the efficiency impact of ambiguity in the parser.

5.6 Parser evaluation and analysis

Table 5.1 summarises our evaluation of the time and space requirements of the MMCCG-enabled version of C&C obtained through the work of this chapter. These evaluations are performed against Section 00 of each version of CCGbank, in line with the usual dataset split first described in Section 3.4.6.

Section 00

Corpus	*Total parsing time*	*Parsing rate*		*Avg. nodes*	
		Sentences/sec	*Words/sec*	*Conj.*	*Total*
Baseline C&C	115.45 sec	16.57	393.43	3398.39	8058.80
All modes	132.44 sec	14.44	342.96	3583.10	8567.29
Without ⋈	**113.09** sec	**16.92**	**401.65**	**3358.70**	**7847.86**

Table 5.1: C&C time and space requirements on selected corpora

Corpus	LP	LR	LF	LF*	LSA	UP	UR	UF	SUP	COV
Baseline C&C	85.53	84.71	85.12	83.38	32.14	92.37	91.49	91.93	93.05	99.06
All modes	82.77	83.60	83.18	81.45	25.33	89.90	90.81	90.35	92.37	98.22
Without ⋈	83.83	84.77	84.30	82.64	25.54	90.49	91.51	91.00	93.03	99.06

Table 5.2: C&C standard parser evaluation on selected corpora

The baseline consists of the unmodified (pure CCG) version of the C&C parser, trained on an unmodified CCGbank. The new evaluations are performed on two versions of the one-to-one mode mapped corpora obtained in Section 4.4: the first carrying the full set of annotations, and the second excluding the impact of the null mode. As shown in Table 5.1, the corpus annotated with the full set of three modes actually yields a parser which underperforms the baseline, while a parser trained on a version of the moded corpus without the null mode exceeds the baseline.

The standard parser evaluation shown in Table 5.2 is identical to that described in Section 3.2.1. We see that the null mode as it is currently specified in the corpus has a negative effect on parser accuracy. Given these marked differences in parser behaviour when we include, or exclude the null mode, we find it prudent to concentrate our analysis on explaining the observed results.

We examined the parser output in greater detail to explain the observed behaviour, to allow us to identify what aspects of the corpus generation methodology should be improved by further research. To understand the impact of parser ambiguity on C&C, we explain the supertagger behaviour which is the key to the parser's efficiency [Clark and Curran, 2007]. The C&C supertagger is controlled by a parameter β: a category

whose probability of selection is within a factor of β of the most likely category is considered. Intuitively, larger values of β yield a smaller, and more focused set of categories. Decreasing β allows the parser to increase its coverage by considering less likely category assignments.

When the parser fails to find a spanning analysis with a given value of β, it can request that the supertagger *decrease* its value of β, producing a larger set of category assignments for the parser to try. In C&C, the successive values of β tried by the parser are fixed: 0.075, 0.03, 0.01, 0.005, 0.001, considered in this order. When no spanning analysis is found at any of these levels of β, the parse fails. It is this tight interface between the parser and supertagger which enables C&C to efficiently handle sentences which involve high-probability category assignments (by only considering less likely assignments when they are needed) without sacrificing coverage or completeness.

Part of the parser output is an analysis of the number of derivations which succeeded on a given setting of β, giving us some insight into the impact of our corpora on parser ambiguity.

	Baseline		*All modes*		*Without* $\bowtie$	
$\beta = 0.075$	1812/1895	(95.62%)	1748/1879	(93.03%)	1812/1895	(95.62%)
$\beta = 0.03$	35/1895	(1.85%)	35/1879	(1.86%)	36/1895	(1.90%)
$\beta = 0.01$	17/1895	(0.90%)	21/1879	(1.12%)	15/1895	(0.79%)
$\beta = 0.005$	8/1895	(0.42%)	11/1879	(0.59%)	8/1895	(0.42%)
$\beta = 0.001$	23/1895	(1.21%)	64/1879	(3.41%)	24/1895	(1.27%)

Table 5.3: β cutoff evaluation on successfully parsed Section 00 sentences

We can see that the *All modes* corpus causes more derivations to fail at $\beta = 0.075$, the first level of β. Also, the number of derivations requiring the supertagger's most permissive level $\beta = 0.001$ has significantly increased. It should be clear that the cost to efficiency incurred by a derivation increases together with how many values of β must be considered before an analysis is found. This impacts not only the parser's time requirements but its space requirements as well, because smaller values of β result in greater categorial ambiguity, and hence larger packed charts. Furthermore, *All modes* actually causes a loss of coverage relative to the other two corpora, with only 1879 derivations covered. As explained before, a parse fails when none of the five levels of β in the supertagger allows the parser to find a spanning analysis. On the other hand, we can see that the corpus *Without* $\bowtie$ has no discernible impact on parser coverage or supertagger utilisation.

We have the following hypothesis for the reduction in coverage on the corpus *All modes* relative to *Without* $\bowtie$. The reduction in coverage on the corpus *Without* $\bowtie$ is due to over-restrictive application of the null mode. From manual inspection of CCGbank,

it is the clausal categories shown in Table 5.4 which exhibit the strongest evidence for assigning the null mode. In each of these cases, the topmost slash in these categories is never, or extremely rarely consumed at all. Furthermore, our knowledge of the structure of CCGbank informs us that these categories are always *introduced* (and hence consumed whole) by some auxiliary verb. Just as Figure 4.9 shows that a past participle is only ever introduced by the forms of the English auxiliary verb *has*, each of the above categories is only ever consumed whole by a word which introduces a clause of the given category. In summary, our knowledge of CCGbank supports the assignment of the null mode to the slashes of these categories. Accordingly, we do not believe that the assignment of the null mode to the slashes of these clausal categories is responsible for the reduction in coverage and the increase in parser ambiguity seen in Table 5.3.

Category	*Comb.*	*Frequency*
$S[adj]\backslash NP$		7035
$S[adj]\backslash NP$	$<$	1
$S[pss]\backslash NP$		6562
$S[pss]\backslash NP$	$<$	5
$S[b]\backslash NP$		2473
$S[b]\backslash NP$	$<$	1
$S[ng]\backslash NP$		1495
$S[ng]\backslash NP$	$<$	3
$S[pt]\backslash NP$		696
$S[asup]\backslash NP$		246
$S[to]\backslash NP$		24

Table 5.4: Slash/combinator analysis for VP categories in Sections 02-21 of CCGbank

Our approach has assigned the null mode to many slashes which do not have this property, on the basis of frequency. To determine whether these other assignments of the null mode are responsible for the loss in coverage on the *All modes* corpus, we would need to undertake a manual analysis similar to the one we have performed to annotate by hand those slashes which require composition. Although we are unable to engage in this analysis as part of the present work, we intend to investigate the impact of the null mode further, in future work. We still believe that the null mode has the potential to reduce, instead of increase, parser ambiguity and efficiency, and we conjecture that restricting the null mode to the slashes of the categories given in Table 5.4 will restore parser efficiency on a corpus incorporating the null mode to at least the levels achieved by the C&C baseline.

5.7 Summary

Our most significant contribution in this chapter is conducting the first evaluation of MMCCG in a wide-coverage statistical parser trained on a large-scale corpus ever performed in the literature, and demonstrating that MMCCG does offer efficiency gains over pure CCG in time and space. In addition, our initial evaluation of MMCCG opens up a number of compelling directions for future research, including further refinement of the moded corpora, and modifications to the supertagger which better utilise its power. Our work has established MMCCG as a viable basis for a wide-coverage statistical parser. We believe that our contribution of the modified C&C parser will not only serve as the basis for our continued work with the multi-modal CCG, but also increase the visibility and appeal of MMCCG as a worthy focus in the field of statistical parser research at large.

CHAPTER 6

Conclusion

The ability to derive deep semantic structure from an increasingly large quantity of natural language information has made efficient, wide-coverage parsing a necessary goal in natural language processing. We have shown that Combinatory Categorial Grammar [Steedman, 2000] is a powerful but concise representation of natural language syntax, which has served as the target for wide-coverage corpora as well as parsers of unsurpassed efficiency. Despite its appeal, we have identified shortcomings in the ability of CCG to describe fine-grained grammaticality distinctions for the prevention of overgeneration, the correct treatment of which would require us to weaken a number of desirable properties otherwise attributable to CCG.

We have shown that pure CCG approaches do not satisfactorily address the overgeneration problem without harming the property of lexicality, requiring a novel approach to add derivational control in a lexicon-driven manner. The refinements to CCG offered by multi-modal Combinatory Categorial Grammar [Baldridge, 2002] successfully address the theoretical concerns of overgeneration without reducing lexicality, while promising improvements in the memory and time consumed by a MMCCG parser relative to a parser built on pure CCG. However, the only validation of MMCCG which exists in the literature is in the form of the manual analyses, and small-scale experiments with a hand-crafted lexicon performed by Baldridge.

To substantiate these claims, we set out to create the first wide-coverage MMCCG corpus and parser in the literature. Our first contribution is the creation of an annotation and corpus processing framework, which allows for the transformation and exploration of large-scale CCG corpora. To demonstrate our ability to interpret, process and transform the derivation structure of CCGbank, we have performed a key canonicalisation task on CCGbank, restoring to it the quote symbols stripped from it during its generation process, presenting our approach to quote reinstatement in Chapter 3. We have shown that the supertagging component of a parser can make use of quotes to inform its assignment of categories, and our conjecture, to be verified by future work, is that increasing the window size to accommodate the insertion of quotes will allow the supertagger to better integrate the presence of quotes into its model. Our contribution of a re-quoted version of CCGbank, encoding a more faithful representation of English, is made available as a general purpose resource for arbitrary corpus applications.

We have applied our corpus annotation framework to the new task of generating various corpora annotated for the MMCCG grammatical formalism. We have surveyed the methodological issues inherent in creating a wide-coverage MMCCG corpus, created a version of the mode system specifically for our work, introduced a novel mode to our mode hierarchy, and conducted a detailed linguistic analysis of the English syntax requiring each of the degrees of combinatory rule freedom in our own mode scheme. Having applied our corpus creation methodology to the wide-coverage CCG corpus CCGbank, we achieve the first wide-coverage MMCCG corpora in the literature, one of the key contributions of this work. We also develop a novel method of further refining the mode annotations – mode splitting – which aims to better utilise the power of the supertagger and parser components, and present the algorithms necessary to support these improvements to the corpus derivation process.

Lastly, we have made the structural extensions to C&C which allow it to interpret and enforce the mode restrictions which are the key contribution of MMCCG. Having obtained the first MMCCG parser capable of being evaluated on the wide-coverage parsing task, we run a full parser evaluation on moded corpora to show that modes contribute positively to the efficiency characteristics of C&C.

The avenues of future work springing from this first investigation and validation of MMCCG in the literature are compelling and diverse. Although most of this work has dealt with corpus manipulation or modifications to the parser component, further investigating the parser-supertagger interface which is one of the unique characteristics of C&C, and exploring the impact of MMCCG in not only reducing the load on the parser but making more efficient use of the supertagger, is a tantalising research direction we intend to continue to pursue. Although further investigation into the power of MMCCG in the supertagger would require a more detailed understanding of the vast C&C codebase beyond the scope of this present work, the techniques and preliminary results of Section 4.5 show that MMCCG has the potential to improve not only the parser but the quality of supertagging as well. Importantly, as we develop finer-grained, more precise moded corpora in future, the corpus transformation and annotation framework created for this work remains a viable way to immediately generate new MMCCG corpora for evaluation.

As the first discussion in the literature of the implications of MMCCG as the foundation of a statistical parser, we open up numerous doors to exciting extensions of the base formalism, motivated by the recognition of MMCCG as a viable basis for efficient wide-coverage parsing. The results obtained in Chapter 5 are a promising sign that the efficiency benefits claimed for MMCCG by Baldridge are indeed attainable. We will continue our investigation into the impact of MMCCG on the modified parser, and we hope that the present work may serve as the first step to the goal of implementing Baldridge's

system in its full generality within the C&C parser.

For the very first time, MMCCG has been lifted from its theoretical setting into the realm of practical, wide-coverage, efficient parsing. Although the milestones of this work – the MMCCG corpus derivation techniques we have developed, the moded corpora themselves, and the embedding of MMCCG in a parser – have been undertaken with the goal of realising the benefits of efficiency and precision attributed to MMCCG, we believe that our work will also serve to encourage further research into the formalism, with the key resources developed through our work independently serving to promote and communicate the numerous advantages to parsing with multi-modal Combinatory Categorial Grammar.

Bibliography

Kazimierz Ajdukiewicz. 1935. Die syntaktische Konnexität. *Studia Philosophica*, 1(1-27).

Lluïsa Astuc-Aguilera and Francis Nolan. 2007. Variation in the intonation of sentential adverbs in English and Catalan. *Tones and Tunes (to appear)*.

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Jason Baldridge and Geert-Jan M. Kruijff. 2004. Course Notes on Combinatory Categorial Grammar. *Unpublished course notes. Available at http://folli.loria.fr/cds/2004/content/readers/51.pdf*.

Yehoshua Bar-Hillel. 1953. A Quasi-Arithmetical Notation for Syntactic Description. *Language*, 29(1):47–58.

Yehoshua Bar-Hillel, Chaim Gaifman, and Eliyahu Shamir. 1960. *On Categorial and Phrase-structure Grammars*. The Weizmann Science Press.

Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics IWCS*, volume 6, pages 42–53.

John Carroll, Guido Minnen, and Ted Briscoe. 1999. Corpus Annotation for Parser Evaluation.

John Chen, Srinivas Bangalore, and K. Vijay-Shanker. 2006. Automated extraction of Tree-Adjoining Grammars from treebanks. *Natural Language Engineering*, 12(03):251–299.

Noam Chomsky. 1993. *Lectures on Government and Binding: The Pisa Lectures*. Walter de Gruyter.

Kenneth Church and Ramesh Patil. 1982. Coping with Syntactic Ambiguity. *American Journal of Computational Linguistics*, 8(3-4):139.

S. Clark and J.R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics.

Stephen Clark. 2002. Supertagging for combinatory categorial grammar. pages 19–24.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Statistical Parsing with CCG and Log-Linear Models. *To appear in* Computational Linguistics 2007. *Unpublished draft manuscript*.

Stephen Clark and Julia Hockenmaier. 2002. Evaluating a Wide-Coverage CCG Parser. In *Proceedings of the LREC 2002 Beyond PARSEVAL workshop*, pages 60–66.

T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. 2001. *Introduction to Algorithms*. The MIT Press, 2nd edition.

James R. Curran, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 697–704. Association for Computational Linguistics.

Haskell B. Curry, Robert Feys, and William Craig. 1958. *Combinatory logic*. North-Holland Publishing Company.

Jason Eisner. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 79–86. Association for Computational Linguistics Morristown, NJ, USA.

Gerald Gazdar. 1985. *Applicability of indexed grammars to natural languages*. CSLI.

L. Gillick and Stephen J. Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-89)*, pages 532–535.

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

Julia Hockenmaier and Mark Steedman. 2001. Generative models for statistical parsing with combinatory categorial grammar. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 335–342. Association for Computational Linguistics, Morristown, NJ, USA.

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

Christopher Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT Press.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-Oriented Grammar Development for Acquiring a Head-Driven Phrase Structure Grammar from the Penn Treebank. pages 684–693.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A comprehensive grammar of the English language*. Longman New York.

John A. Rice. 1995. *Mathematical statistics and data analysis*. Duxbury Press.

Yves Schabes, Anne Abeille, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: application to tree adjoining grammars. In *Proceedings of the 12th conference on Computational linguistics*, pages 578–583. Association for

Computational Linguistics, Morristown, NJ, USA.

Stuart M. Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343.

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

Ann Taylor, Mitchell P. Marcus, and Beatrice Santorini. 2003. The Penn Treebank: an overview. *Treebanks: Building and Using Parsed Corpora*, pages 5–22.

University of Chicago Press. 2001. Chicago Manual of Style Online. URL http://www.chicagomanualofstyle.org, Accessed 16 September 2007.

K. Vijay-Shanker and D.J. Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

David J. Weir and Aravind K. Joshi. 1988. Combinatory Categorial Grammars: generative power and relationship to Linear Context-Free Rewriting Systems. pages 278–285. Association for Computational Linguistics Morristown, NJ, USA.