Creating a Natural Logic Inference System with Combinatory Categorial Grammar

Christos Christodoulopoulos

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2008

Abstract

This dissertation presents an integrated system for producing Natural Logic inferences, which are used in a wide variety of natural language understanding tasks. Natural Logic is the process of creating valid inferences by making incremental edits to natural language expressions with respect to a universal monotonicity calculus, without resorting to logical representation of the expressions (using First Order Logic for instance).

The system generates inferences from surface forms using a three-stage process. First, each sentence is subjected to syntactic analysis, using a purpose-built syntactic parser. Then the rules of the monotonicity calculus are applied, specifying the directionality of entailment for each sentence constituents. A constituent can be either upward or downward entailing, meaning that we may replace it with a semantically broader or narrower term. Finally, we can find all suitable replacement terms for each target word by using the WordNet lexical database, which contains hypernymic and hyponymic relations.

Using Combinatory Categorial Grammar, we were able to incorporate the monotonicity determination step in the syntactic derivation process. In order to achieve wide coverage over English sentences we had to introduce statistical models into our syntactic parser. At the current stage we have implemented a simple statistical model similar to those of Probabilistic Context-Free Grammars.

The system is intended to provide input to "deep" reasoning engines, used for higher-level Natural Language Processing applications such as Recognising Textual Entailment. In addition to independently evaluating each component of the system, we present our comprehensive findings using Cyc, a large-scale knowledge base, and we outline a solution for its relatively limited concept coverage.

Acknowledgements

I would like to begin by thanking Mark Steedman. As a lecturer, he introduced me to the wonderful subject of computational linguistics, presenting it with great enthusiasm through his own experience. At the same time, his lectures were the best initiation to my MSc and my experience of the UK higher education in general. Even before he became my supervisor, he was always supportive and keen to examine new ideas that eventually became the main themes of this project. His suggestions and comments were precise, guided by his broad knowledge and insight. And yet he was always accessible, ready to discuss my proposals and ideas. In our short collaboration Mark has been a real source of inspiration for me.

I would also like to thank Jason Baldridge from the University of Texas at Austin. His help with the technical details of OpenCCG was essential to the creation of my statistical parser and when I got the opportunity to meet him in person, his advice and comments were equally helpful.

I'm very grateful to the faculty and students of Computational Linguistics at Ohio State University who made the first effort in creating a statistical version of the OpenCCG parser. Especially, I would like to thank Dennis Mehay, as he was kind enough to let me use his Suppertagger code as a starting point for my Supertagger, but also for his great technical advise.

Finally, I would like to express my gratitude to Emily Thomforde from the University of Edinburgh. Our meetings were always a source of encouragement and inspiration for me. She became not only my linguistics *liaison* but also a wonderful advisor and friend. She helped me get through many tough spots, by being so supportive and enthusiastic about my project. During the last few weeks she read tirelessly through all of my drafts correcting my imperfect English and inconsistencies. Thank you for everything Emily.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Christos Christodoulopoulos*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

> . . . a logic for natural language, whose goals are to express all concepts capable of being expressed in natural language. . .
> George Lakoff, 1970

In this project we will examine the creation of a Natural Logic inference system in terms of both the theoretical concepts and the practical issues that are involved. Although Natural Logic, in the form of logical inference over natural language expressions, can be traced back to Aristotle, it was not until the end of the 20th century, where it was formalised into a *monotonicity calculus* by Sánchez-Valencia (1991, 1995), that Natural Logic could be used as an inference system.

Such a system would be able to construct proofs based on incremental edits of natural language expressions. In other words, one could derive a number of entailments from a "surface structure" sentence without resorting to a formal logical analysis and without changing the truth of the sentence (*salva veritate*). This is made possible by the identification of certain words within the sentence that define an entailment *directionality*. The sentence could be either upward or downward entailing with respect to that word, which in turn specifies whether the word in question can be replaced by a more general or a more specific one respectively.

It is the purpose of the Natural Logic theory to specify how this directionality can be calculated and it is up to the system to be able to use this information to create the entailment expressions. The purpose of this project is therefore to create such a system that can serve as a platform for the theory of Natural Logic. Unlike previous attempts, this project will emphasise the creation of a *wide-coverage* tool that will be able to deal with real-life texts and, combined with a comprehensive knowledge base could improve the accuracy of knowledge-based Natural Language Processing (NLP) tasks,

such as Word Sense Disambiguation, Textual Entailment or Question Answering.

In fact one of the main motivations behind this project was the relatively poor performance of large-scale knowledge-based systems (in this case *Cyc*) on NLP tasks as portrayed in Mahesh et al. (1996) and more recently in Cox (2005). Both reports emphasise that the main problem with Cyc was its poor lexical coverage and the low interconnection between concepts. Our intuition here is that we can use Natural Logic entailments as a means to rectify Cyc's (or other systems') lack of specific concepts. In particular we can use the Natural Logic system to provide us with synonymous or more general concepts when the knowledge base fails to find a relevant match for a given word/concept.

The creation of this system is a multi-stage task involving the processes of natural language wide-coverage parsing with Natural Logic polarity marking as its base; word sense disambiguation, synonym/hypernym retrieval and logical inference will be the higher level extensions of the base towards more sophisticated NLP applications.

Given the time limitations of the project, we will focus primarily on the creation of the base system as a proof-of-concept for the Natural Logic theory in a wide-coverage environment but we will also address some of the issues concerning the interface between our system and the higher-level procedures (i.e. the knowledge base and the model builders/theorem provers).

The initial results show that wide-coverage Natural Logic inference is possible, although we don't have enough evidence to suggest that this is sufficient to provide a significant improvement on the higher-level applications such as Textual Inference. Nevertheless our results show that we can obtain greater concept coverage from knowledge-bases like Cyc, which is indicative of better results in such applications as both Mahesh et al. (1996) and Cox (2005) propose.

In chapter 2 we examine the theoretical background of the concepts involved in this project, while in chapter 3 we review the related work in each of the processes involved, as well as in the creation of other Natural Logic inference systems. Chapter 4 presents the details of the system and the key points of some of the proposed extensions and chapter 5 presents the results of the various parts of the system and an empirical evaluation of the system as a whole. Finally in chapter 6 we close with a discussion of the current limitations of the system and a description of the possible directions of the research.

Chapter 2

Background

2.1 Natural Language Parsing

The first step in any Natural Language Processing application is the analysis of a given sequence of tokens (a sentence, a text etc.) to determine its syntactic and consequently its semantic structure[1]. A parser is a system that performs this analysis and consists of a formal grammar that specifies the list of all the possible constituents of the language and their possible analyses, and an algorithm that applies the rules of the grammar to the constituents. The grammar formalism that we will use is the Combinatory Categorial Grammar presented in the next section.

2.1.1 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG; Steedman (1996, 2000)) is a grammar formalism that generalises the Categorial Grammar family of Ajdukiewicz (1935) and Bar-Hillel (1953). Categorial Grammar (CG) is one of the first lexicalised grammar formalisms in that all the constituents of the grammar are distinguished by a type (or *category*) that identifies them either as a function from categorial arguments to results or as an argument.

(2.1) a. John := NP

b. walk := $S\backslash NP$

c. see := $(S\backslash NP)/NP$

d. that := $(NP\backslash NP)/(S[dcl]/NP)$

[1]These do not have to be separate derivations. The grammar formalism that we are using (CCG) allows the semantic structure to be derived directly from the surface form Steedman (2000)

The function category in 2.1c identifies the transitive verb as function and specifies the type of its arguments and results. We use here the "result leftmost notation" in which the range of the functor α is written before the slash and the domain β is written after it. The slashes specify the directionality of the arguments with / corresponding to a leftward-combinating functor and \ corresponding to a rightward-combinating one. Finally as we can see from example 2.1d that both α and β can be function categories themselves.

Therefore CGs rely on their lexicon instead of using phrase structure rules to capture the syntactic information of the grammar constituents. In its original form CG was limited to *functional application* combinatory rules and was proven of being weakly equivalent to context-free phrase-structure grammars (CFPSGs) and therefore having context-free expressive power[2] (Bar-Hillel et al., 1964).

(2.2) Functional Application:

a. $X/Y \quad Y \quad \Rightarrow \quad X \quad (>)$

b. $Y \quad X\backslash Y \quad \Rightarrow \quad X \quad (<)$

CCG generalises pure Categorial Grammars, to include further operations on categorial functions that are related to the combinators identified by Curry and Feys (1958) namely Composition(**B**), Type-Raising(**T**) and Substitution(**S**). These rules were introduced in order to capture non-context free phenomena like crossing predicate-argument dependencies in Dutch and related Germanic Languages, the coordination of non-standard contiguous constituents and unbounded dependencies.

[2]Context-free grammars is one of the grammar classes in *Chomsky's Hierarchy*, a containment hierarchy of classes of formal languages. In its original form (Chomsky, 1957) the hierarchy contained 4 categories of grammars with every category being a proper subset of the type above it:

Language Type	**Grammar**	**Automaton**
Type-0	Recursively enumerable	Universal Turing machine
Type-1	Context-sensitive	Linear Bound Automaton
Type-2	Context-free	Push-Down Automaton
Type-3	Regular	Finite-State Automaton

(2.3) Functional Composition:

a. $X/Y \quad Y/Z \quad \Rightarrow_{\mathbf{B}} \quad X/Z \quad (>\mathbf{B})$

b. $X/Y \quad Y\backslash Z \quad \Rightarrow_{\mathbf{B}} \quad X\backslash Z \quad (>\mathbf{B}_{\times})$

c. $Y\backslash Z \quad X\backslash Y \quad \Rightarrow_{\mathbf{B}} \quad X\backslash Z \quad (<\mathbf{B})$

d. $Y/Z \quad X\backslash Y \quad \Rightarrow_{\mathbf{B}} \quad X/Z \quad (<\mathbf{B}_{\times})$

(2.4) Functional Substitution:

a. $(X/Y)/Z \quad Y/Z \quad \Rightarrow_{\mathbf{S}} \quad X/Z \quad (>\mathbf{S})$

b. $(X/Y)\backslash Z \quad Y\backslash Z \quad \Rightarrow_{\mathbf{S}} \quad X\backslash Z \quad (>\mathbf{S}_{\times})$

c. $Y\backslash Z \quad (X\backslash Y)\backslash Z \quad \Rightarrow_{\mathbf{S}} \quad X\backslash Z \quad (<\mathbf{S})$

d. $Y/Z \quad (X\backslash Y)/Z \quad \Rightarrow_{\mathbf{S}} \quad X/Z \quad (<\mathbf{S}_{\times})$

(2.5) Type-raising:

a. $X \quad \Rightarrow_{\mathbf{T}} \quad T/(T\backslash X) \quad (>\mathbf{T})$

b. $X \quad \Rightarrow_{\mathbf{T}} \quad T\backslash(T/X) \quad (<\mathbf{T})$

The new rules increased the expressivity of CCG above context-free power. More specifically it was shown by Vijay-Shanker and Weir (1994) that CCG was weakly equivalent to Head-driven Phrase Structure Grammar (HPSG; Pollard and Sag (1994)), Tree Adjoining Grammar (TAG; Joshi (1988)) and Linear Indexed Grammar (LIG; Gazdar (1988)). Furthermore Vijay-Shanker and Weir showed that all these formalisms delineated a new level in Chomsky's Hierarchy (see footnote 2): *mildly-sensitive grammars*, which is suggested to be the lowest level in which all syntactic phenomena of natural grammar can be captured.

However, these rules also introduce the phenomenon of *spurious ambiguity*. Using type-raising and the other combinatory rules, CCG can generate "non-standard" constituents. This ability creates large amounts of logically equivalent but syntactically distinct derivations. In fact in long strings these "extra" parses can be up to a Catalan number[3].

Eisner (1996) proposes a solution for this problem using a *normal-form* parsing technique. A parse is said to be in normal form when it satisfies the following constraints:

[3]The Catalan number for a sentence with n words is $\frac{(2n)!}{(n+1)!n!}$

(2.6) a. No constituent produced by $>\mathbf{B}n$, any $n \geq 1$, ever serves as the primary (left) argument to $>\mathbf{B}n'$, any $n' \leq 0$.

b. No constituent produced by $<\mathbf{B}n$, any $n \geq 1$, ever serves as the primary (right) argument to $<\mathbf{B}n'$, any $n' \leq 0$.

Put more simply, these constraints say that the output of rightward (respectively leftward) composition may not compose or apply over anything to its right (respectively left).

The syntactic categories in CCG include agreement features as subscripts (e.g. NP_{3sf} and $S\backslash NP_{3s}$), but most of the time this information will be omitted as it is of little relevance to this project. Apart from the syntactic information of the constituents, CCG categories also capture semantic interpretations which can be thought of either as predicate-argument structures or as purely model-theoretic objects[4]. A common practice in Steedman (1996) and thereafter is to represent the interpretations as predicate-arguments structures by associating a λ-term with the syntactic category via a colon operator. Following the Principle of Combinatory Type-Transparency (Steedman, 2000) all the rules must be expanded in a way that the semantic type of the reduction is the same as its syntactic type. Example 2.7 presents a simple CCG derivation that produces a semantic interpretation in the form of λ-calculus.

(2.7)
$$\dfrac{\dfrac{\dfrac{\textit{John}}{\mathsf{NP} : \mathsf{john}'} \quad \dfrac{\textit{sees}}{(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.\mathsf{see}'xy} \quad \dfrac{\textit{Mary}}{\mathsf{NP} : \mathsf{mary}'}}{\mathsf{S}\backslash\mathsf{NP} : \lambda y.\mathsf{see}'\mathsf{mary}'y}\,{>}}{\mathsf{S} : \mathsf{see}'\mathsf{mary}'\mathsf{john}'}\,{<}$$

It is this "natural" interpretation of semantics that makes Categorial Grammars and CCG in particular an ideal candidate for this project. "Natural" is used here in the sense that CCG provides a direct and transparent interpretation of the surface structure which means that we can apply Natural Logic's rules of monotonicity while parsing the surface structure. We will discuss this further in chapter 4.

An expansion of CCG, deviating from Steedman (2000), was introduced by Baldridge (2002) and Baldridge and Kruijff (2003). *Multi-Modal* CCG restricted the function categories as to the rules that allow them to combine with other categories, via "modalised"

[4]CCG can also incorporate intonation information into categories. These features can be used to identify the *theme* and *rheme* structures in a sentence which can be useful in question-answering applications. For further information about intonational structure representation in CCG, see Steedman and Prevost (1994). In this case CCG's spurious ambiguity is in fact helpful.

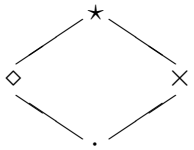


Figure 2.1: CCG slash modalities hierarchy (adapted from Baldridge and Kruijff 2003).

slashes. The slash modalities are a set of four feature values $\{\star,\times,\diamond,\cdot\}$ and their hierarchical relation is represented in figure 2.1. This hierarchy can be interpreted as follows: the $\star$ type is the most restricted and allows only the most general applicative rules (functional application and coordination[5]), $\diamond$ permits order-preserving associativity, $\times$ allows limited permutations and $\cdot$ is the most permissive slash modality and allows all the rules to apply.

The slash modalities have an important effect on CCG's combinatory rules, permitting the lexical categories to have control over them. This allows CCG rules to become truly universal in that the grammar of every language will use exactly the same set of rules, leaving all the language specific restrictions and bans to be defined in the lexicon. For a more comprehensive presentation of this expansion as well as a complete introduction to CCG see Steedman and Baldridge (2007).

2.1.2 Wide-Coverage Parsing

Having discussed the grammar formalism we turn to examine the theoretical concepts and algorithms of wide-coverage parsing. Parsing algorithms can be characterised by one criterion discriminating the way the parser constructs the derivation structures called *parse trees*. With respect to whether the parser starts the construction from the input sentence or from the grammar's starting symbol[6], we have *bottom-up* and *top-down* parsers (Roeck, 1983).

However, because of the number of the ways in which multiple parses for different phrases can be combined, both bottom-up and top-down approaches in their pure form are ineffective. For this reason algorithms that use the notion of *dynamic programming*

[5]While in previous version of CCG theory (Steedman, 2000) coordination is treated using a special rule and not with a $(X\backslash X)/X$ category as it would overgenerate, the modalised version $(X\backslash_\star X)/_\star X$ is acceptable. We examine coordination more thoroughly in section 4.1.4.

[6]This is the symbol that corresponds to the *root* of the tree in any linguistic structure built; usually it is the symbol **S**.

INITIALISATION:
for $(i = 0$ **to** $chart.size)$
  insert next constituent in $chart[i][i]$

CHART PARSING:
for $(j = 1$ **to** $chart.size)$
  for $(i = j-1$ **to** $i = 0, i--)$
    for $(k = i$ **to** $k < j)$
      unify cell $chart[i][k]$ with cell $chart[k+1][j]$
      insert derivation in $chart[i][j]$
    unify cell $chart[i][j]$ (unary expansions)

Figure 2.2: The Cocke-Kasami-Younger algorithm (adapted from Hockenmaier, 2003)

have been suggested. The basic idea of dynamic programming is that each time we parse a substring we can store the results so that we won't have to re-analyse it again. For the parsing algorithms this idea is embodied in a data structure called a *chart*. The chart for a sentence is an $n \times n$ half-table where n is the length of the sentence. Each cell in the chart ($chart[i][j]$) contains a list of constituents that span a substring of the original sentence from word w_i to word w_j. Russell and Norvig (2003, pp. 800–806) offer a very comprehensive introduction to chart parsing.

While there are many parsing algorithms of both types in the related literature, here we will focus on the Cocke-Kasami-Younger (CKY) chart parsing algorithm as this the one we will be using in the project.

CKY in its original form was developed for context-free grammars (Younger, 1967) which have a finite number of nonterminal categories. This allowed the CKY to achieve polynomial parsing time of $O(n^3)$. However, given that the set of categories in CCG can theoretically be infinite (Steedman, 2000), parsing in the original implementation of CKY might reach exponential times. An algorithmic solution has been proposed by Vijay-Shanker and Weir (1993)[7] that bounds the parsing time to $O(n^6)$ and although their method is quite effective, in this project we will overcome the problem using a more practical approach. As shown in Hockenmaier (2003), if we constrain the set of CCG categories for the probability models to those that occur in our train-

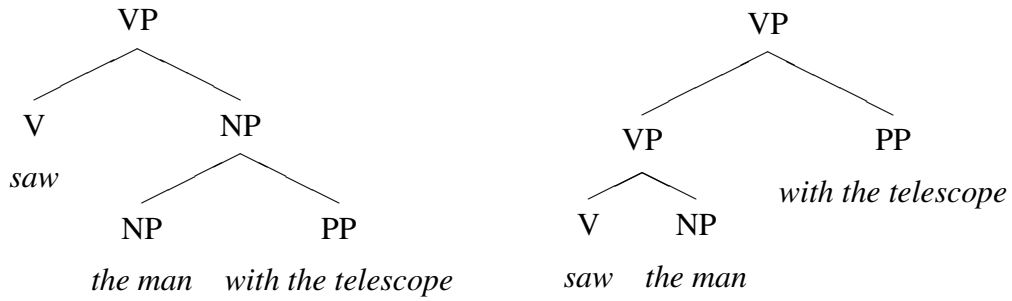
[7] What they propose is a structure-sharing technique that encodes complex categories whose *arity* is greater that a fixed bound. As this is only an overview of the methods used in the project we will not proceed into further analysis. See Vijay-Shanker and Weir (1993) for more details.

ing data, we effectively bound the maximum number of the categories in parser and therefore reduce the complexity back to polynomial time. We will further discuss this approach in chapter 4.

The CKY algorithm provides a basis for the syntactic analysis of a sentence given the grammar (lexicon and rules), and we could say that it completes the most important part of the parsing process. However wide-coverage parsing has a third element that has an equally import role. This is the statistical model and we will now discuss some of its aspects.

But first let's consider why is it important to use statistical parsing in the first place. The answer lies in the fact that most everyday sentences are ambiguous with respect to their syntactic (and consequently their semantic) analyses. Sentence 2.8 presents a simple example where the Prepositional Phrase **with the telescope** can modify either the noun **man** or the verb **saw**.

(2.8) John saw the man with the telescope.



This of course has an impact on the semantic analysis of the sentence where in the first case the telescope is used to discriminate the man whereas in the latter analysis the telescope is used as an instrument for seeing. With syntactic ambiguity growing significantly with the length of the sentence, it's not difficult to understand why most of our real-life sentences are highly ambiguous[8]. The statistical model is used in parsing as an oracle that deals with the ambiguity of the sentence.

As syntactic parsing is but an intermediate step in our analysis, we normally want to keep the best (or the most likely) parse. More formally we want the parse with the highest *likelihood* among all the possible parses (τ) of a sentence string (α) given the grammar G.

$$\tau^* = arg\max_{\tau} P(\tau|\alpha) \tag{2.9}$$

[8]Manning and Schütze (2000) using a Wall Street Journal article as source for their examples, show that even the second (10-word) sentence of the article has at least 5 well-formed syntactic structures.

Before we examine any specific statistical model we should look examine two different approaches to statistical parsing: *Generative* and *discriminative* models. Although in theory both models can be used to retrieve the conditional probability $P(\tau|\alpha)$, a generative model is full probability model of all variables while a discriminative model can be used only as a model of the target variable τ, conditional on the given variable α. Therefore a generative model can be used to simulate (or generate) values of any variable in the model, making it the ideal model to describe *generative grammars*.

More important for the present purpose is their behaviour within a syntactic parser and more specifically the way each model handles new words in a bottom-up parser. Here the discriminative model makes a parsing decision for each new word using *lookahead* to provide unseen words as input for the current decision. The generative model in contrast predicts each word only when it reaches the parse, thus delaying the parsing decision until it has more information about the context in which the word appears. In this way generative models can achieve higher parsing accuracy, while the discriminative models are considerably faster (Henderson, 2003).

2.2 Natural Logic

The roots of natural logic can be traced back to Aristotle's Syllogistic which could be described as a logic system based on common nouns and verb phrases. As Sánchez-Valencia (1991) observes many rules in the Aristotelian logic can be described as inferences that replace one predicate with another denoting a superset (or a subset) of the first.

In other words, we can say that natural logic inference is based on a monotone replacement of predicates in natural language expressions. This notion of *monotonicity* captures the fact that certain terms can be either expanded or contracted without changing the truth of the sentence (*salva veritate*). Put more formally, these terms can either be *upward* or *downward* entailing. Monotone entailments preserve or reverse the relation of semantic strength among expressions. Under normal circumstances the relation is preserved and we can entail 2.11 from 2.10 but not 2.12.

(2.10) John walks.

(2.11) $\vdash$ John moves.

(2.12) $\nvdash$ John walks fast.

However there are downward monotone expressions which reverse the relation of semantic strength between the expressions. This means we can only replace the term in question with a more specific (or narrow) sense. Therefore we can't replace 2.13 with 2.14 but only with 2.15.

(2.13) John doesn't walk.

(2.14) $\nvdash$ John doesn't move.

(2.15) $\vdash$ John doesn't walk fast.

To understand how can we obtain the monotonicity (or the directionality of entailment) for each expression we must use a formal system (or *calculus*). Sánchez-Valencia (1995) describes such a *monotonicity calculus* of Natural Logic using the proof system of his Ph.D. thesis (Sánchez-Valencia, 1991). The calculus is based on the following 4 definitions:

Definition 1 *Given two functions f,g we say that $f \leq g$ iff $\forall$ x in their domain it holds that $f(x) \leq g(x)$.*

Definition 2 *Function f is UPWARD MONOTONE if $\exists$ x,y in its domain such that $x \leq y$ it holds that $f(x) \leq f(y)$.*

Definition 3 *Function f is DOWNWARD MONOTONE if $\exists$ x,y in its domain such that $x \leq y$ it holds that $f(y) \leq f(x)$.*

Definition 4 *Function f is non-monotonic if it is neither upward or downward monotonic.*

These definitions allow us to formally prove the intuitions behind the previous examples. We can observe that every natural sentence environment, in the absence of downward monotone expressions, is upward monotone. Therefore in the upward monotone environment of example 2.10 (Jonh [walks]), since $[\![\textbf{walks}]\!] \leq [\![\textbf{moves}]\!]$ is true we can entail "John [moves]" but not "John [walks fast]" (since $[\![\textbf{walks fast}]\!] \leq [\![\textbf{walks}]\!]$).

On the other hand in ex. 2.13 **doesn't** is a downward monotone function and therefore since $[\![\textbf{walks fast}]\!] \leq [\![\textbf{walks}]\!]$ we can entail "John [walks fast]" but not "John [moves]".

We may say that a linguistic intuition for this phenomenon is that, in an upward monotone environment, any predicate α can be replaced by a semantically "smaller" (or narrower) predicate β; respectively α can be replaced by a wider β in a downward monotone environment. Note that, as Sánchez-Valencia (1995) points out, this *inclusion relation* holds between a variety of syntactic categories of which verb phrases are just one. We can also have this relation

- among common nouns; e.g. $[\![\textbf{farmer}]\!] \leq [\![\textbf{man}]\!]$.
- among noun phrases; e.g. $[\![\textbf{John}]\!] \leq [\![\textbf{some man}]\!]$.
- among connectives; e.g. $[\![\textbf{and}]\!] \leq [\![\textbf{or}]\!]$.
- among determiners; e.g. $[\![\textbf{every}]\!] \leq [\![\textbf{a}]\!]$ or $[\![\textbf{more than 4}]\!] \leq [\![\textbf{more than 2}]\!]$.

In larger sentences we can have both directions of entailments with certain words signaling the change of directionality. For instance if we consider the sentence *Every man in France is having wine* we can see that *France* can be narrowed to *Paris* while *wine* could be expanded to *drink*. The difference in the monotonicity of the two nouns lies with the determiner *every* which specifies a downward monotone environment for its adjacent noun.

Using these hypernym/hyponym replacements we can construct a number of logical entailments in the form of the original sentence.

2.3 Word Sense Disambiguation

After parsing the sentence and applying the monotonicity rules the system must be able to replace every entailing word with a more general or more specific concept depending on the direction of the entailment. This requires the use of an electronic lexical resource that contains hyper- and hyponymic relations. For this project we chose WordNet (Fellbaum, 1998) one of the most comprehensive and commonly used lexical databases.

However there is a problem; WordNet's distinctions between the various senses of a certain word are very fine-grained. This problem is even more apparent in everyday sentences as more common words tend to have more senses[9]. This means that in order to able to recognise which of the possible senses of a given word is needed

[9] In fact according to Agirre and Edmonds (2007a) the 121 most frequent English nouns have on average 7.8 senses each.

(to be replaced by its hyper/hyponyms) we will have to use some method of sense disambiguation. The following is a short introduction to this field.

Ever since the late 40s with the first attempts of processing natural language using electronic computers, the task of disambiguating word senses was apparent. *Lexical Ambiguity Resolution* or *Lexical Disambiguation* can be defined as the problem of determining the correct sense of a lexically ambiguous word. In a more restricted point of view –that is applicable to NLP– the task of Lexical Ambiguity Resolution is defined as (Edmonds, 2006): "to computationally determine which sense of a word is is activated by the use of the word in a particular context". Within this context another term is commonly used especially in the last decade: *Word Sense Disambiguation* (WSD), which will be the preferred term in our discussion further on.

Although in WSD literature ambiguity can be either syntactic (which has been discussed earlier) or semantic, here we assume that the parser has resolved successfully the syntactic ambiguity. We will therefore focus on semantically ambiguous words.

Semantic ambiguity can be distinguished into two types: *polysemy* and *homonymy* (Small et al., 1988). Polysemy (from the Greek πολυσημια=having many signs) refers to a word that has several related meanings. For example the uses of **fell** in *The city fell to enemy forces* and *John fell and hurt his knee* are quite similar in terms of meaning but not literally the same. Homonymy (Greek ομωνυμια=having the same name) refers to a word whose various meanings are unrelated (e.g. the two meanings of the word **ball** in *John danced at the ball* vs. *John threw the ball very far*)[10].

Of the two forms of ambiguity, homonymy is considered the easiest to resolve because although polysemy might be more informative, the various senses of a polysemous word tend to be highly indistinguishable. Since this project has to deal with high level semantic decisions (as it will feed a sophisticated knowledge base) we must confront both types of ambiguity. However, as it will become apparent during the literature review, the processes of WSD and (knowledge-based) logical inference are highly interconnected.

[10]This is an oversimplification; there are three more fine-grained distinctions of semantic ambiguity (types of "rhetorical transference"): metaphor, metonymy and synecdoche. Their definitions according to Halliday (1994) are:
In *Metaphor* "a word is used for something resembling that which usually refers to"
In *Metonymy* "a word is used for something related to what it usually refers to"
In *Synecdoche* "a word is used for some larger whole of which that which it refers to is a part"
Although Hirst (1987) suggest that metaphor is not easily separable from polysemy, we will eschew these cases of semantic ambiguity as they are far from being easy to determine and resolve.

Chapter 3

Literature Review

3.1 Natural Language Parsing

Although the related work in the field of natural language parsing (statistical or otherwise) is abundant with examples starting from the mid-70s, there are but a few that are designed specifically for the CCG formalism. For wide-coverage parsing in particular there exist –to our knowledge– only two systems. However, in the interest of completeness, we should mention some of the most significant approaches to parsing natural language.

Woods (1973) offers one of the first parsing systems that is based on the work of Chomsky (1957) in *Transformational Grammar*. Woods implemented the *Augmented Transition Network* (ATN) grammar which extends the *push down automaton* (PDA)[1]. In essence the transition networks were diagrams with states (or nodes) representing the nonterminal grammar constituents and arcs representing the rules of the grammar. ATN extended this network with register-setting actions, most important of which was the *HOLD* action. HOLD dealt with the Transformational Grammar's non-monotonic movements[2] and allowed the parsing of arbitrary deeply embedded constructions.

Woods' parser was used in a question answering system about the chemical analysis of the lunar rock samples brought back from the Apollo missions, at the NASA Manned Spacecraft Center in Houston. It is reported that the system was able to successfully parse and answer 80% of the questions asked within the relevant scope of the data and "another 10% would have been successful but for trivial dictionary and

[1] PDA was a finite-state transition diagram with the context-free expressive power.
[2] More specifically with the phenomenon of *left extraposition*

coding errors"[3].

3.1.1 Wide-Coverage Parsing

While Woods showed that it is possible to create parsers capable of handling highly expressive input, most of the work since, has been concerned with context-free formalisms. This included the research in statistical parsing which monopolised the natural language parsing scene from the mid-90s.

Before we continue, it is important highlight a division in probabilistic parsers (or more accurately their models). We can distinguish between *lexicalised* parsers that take into consideration the words of the sentence as they make their decisions and *non-lexicalised* that deal only with word categories.

Another important aspect in this discussion is the evaluation of the parser. The most prominent measures of evaluation are the PARSEVAL measures proposed by Black et al. (1991). Although originally designed for non-statistical parsers, they have been used extensively as evaluation measures for statistical systems as well. The two most important PARSEVAL metrics are *labeled precision* and *labeled recall*. Equations 3.1 and 3.2 show how these numbers are calculated. The subscripts *ps* and *gs* signify the total number of constituents proposed by the parser and the gold standard[4] respectively.

$$P = \frac{correct\_constituents}{constituents_{tot\_ps}} \tag{3.1}$$

$$R = \frac{correct\_constituents}{constituents_{tot\_gs}} \tag{3.2}$$

One of the most successful early attempts at statistical parsing was made by Charniak (1996) using a non-lexicalised PCFG model to parse the Penn Treebank. The model induced the probabilities estimating the *maximum likelihood* of the parse trees by using the relative frequencies of the local (training) trees. No smoothing or backing-off techniques were used.

Surprisingly Charniak's parser performed extremely well even for today's standards. It achieved 80.4% recall and 78.8% precision. Compared to the state-of-the-art performance ($\sim$ 90%) these results show the power of simple unlexicalised models. There are two possible explanations for this performance. First, while it is true that

[3] Although this may seem like a strikingly good performance even for today's standards, we must keep in mind that the ATN system was able to parse only a fraction of the English grammar and only those sentences that were directly related to system's data. A similar example is that of Winograd (1972)

[4] The term *gold standard* refers to the manually created parse of a sentence in a linguistic corpus.

difficult parsing decisions (like attachment ambiguity) require the use of lexical and/or semantic information, most of the decisions the parser will confront are quite mundane and can be handled adequately by an unlexicalised model. Another possibility is that although some *local trees* (trees of depth 1) are so rare and appear only in the test set, it is unlikely that they would occur in the highest probability parse of even a smoothed lexicalised model.

We should now focus our attention to CCG statistical parsers but first, we must examine the work of Collins (1997) as it is the basis for wide-coverage CCG model of Hockenmaier (2003), which in turn served as a starting point for this project's statistical model. Collins proposed three lexicalised generative models for context-free grammars but he also suggested the use of a *beam search*[5] and smoothing techniques to achieve optimal results.

Another important factor here is that Collins' parser depended on an external part-of-speech tagger (Ratnaparkhi, 1996) that was used to assign part-of-speech tags to unknown words in the test data. This idea is the grounding point for the use of a *supertagger* described in the next chapter.

Collins' models calculate the probability of a parse by iteratively calculating the probability of each local tree in a top-down approach. The probability of each tree (except for the topmost one) is the product of the probabilities of generating a head child $P_{\mathcal{H}}$ and the ith left and right non-head sisters, given a parent node of a tree. Equation 3.3 describes the fist and simplest statistical model of Collins (1997), where $\mathcal{H}$ is the head child, h is the $\langle$word, part-of-speech tag$\rangle$ pair of $\mathcal{H}$, $\mathcal{P}$ is the parent node and $\mathcal{R}$ ($\mathcal{L}$) is the right (left) non-head child with r (l) being its $\langle$word, part-of-speech tag$\rangle$ pair.

$$P_{\mathcal{H}}(\mathcal{H}|\mathcal{P},h) \times \prod_{i=1}^{n+1} P_{\mathcal{R}}(\mathcal{R}_i(r_i)|\mathcal{P},h,\mathcal{H}) \times \prod_{i=1}^{n+1} P_{\mathcal{L}}(\mathcal{L}_i(l_i)|\mathcal{P},h,\mathcal{H}) \tag{3.3}$$

The next model extends the first one by introducing the notion of *subcategorisation* frames. This allows the model to distinguish between complements and adjuncts (e.g. temporal modifiers). The main difference in the calculation of the probabilities is that the non-head children's probabilities are now conditioned on the their subcategorisation lists as well.

Finally the third model incorporates a gap-passing mechanism to deal with *wh-movement* and *traces*. The mechanism was borrowed from grammar formalisms similar to GPSG (Gazdar, 1988), where a *gap* feature is added to every non-terminal node

[5]Beam search is a pruning strategy designed to reduce the size of the search space.

in the tree and the gaps are propagated through the tree, until they are discharged by a trace complement. The calculation of probabilities is again re-defined as the heads now also generate the gap-passing parameter that specifies which node the *gap* feature is passed on to.

We will examine the ideas behind these two models in greater detail when we describe the work of Hockenmaier (2003) in generative models for CCG.

3.1.2 Wide-Coverage Parsers for CCG

3.1.2.1 Generative models

In this section we will examine a number of generative models for CCG proposed by Hockenmaier (2003) and Hockenmaier and Steedman (2002). However in order to better evaluate the results of this model (and the discriminative model of the next section) we must identify some of the differences that separate a wide-coverage CCG parser from the various context-free parsers (and models) that we reviewed earlier.

First and foremost, CCG produces only unary and binary branching trees which means that a direct comparison to gold standard trees (that can be of any arity) is not possible. Furthermore, CCG uses a very fine-grained set of categories, much larger than the standard Penn Treebank part-of-speech set[6] but it requires only a very small set of rules (unary and binary schemata such as function application and composition; see section 2.1.1).

The result of these differences is twofold: first, the original Treebank had to be translated into a CCG-style format[7] and second, the use of the PARSEVAL measures will be insufficient for the CCG parsers. Instead, Hockenmaier and Steedman (2002) propose the use of a dependency evaluation metric which counts word-word dependencies. More specifically, for a local tree with a parent node P, a head child H and a non-head child S this metric counts the dependencies between the head word of H (w_H) and the head word of S (w_S). Using an unlabeled dependency metric –where the label of the local tree that defines the dependency is not used– Hockenmaier and Steedman (2002) were able to compare their scores to those of any other grammar formalism (containing any type of labels and any kind of trees).

The first and simplest of these models is described in figure 3.1. It is an unlexicalised CCG adaptation of the first model in Collins (1997). Despite its simplicity it

[6] Just for reference Hockenmaier and Steedman (2002) report a lexicon of 1207 categories, compared to 48 POS tags of the Penn Treebank.

[7] The creation of this translated *CCGbank* is described in Hockenmaier (2003)

given tree node N with CCG category C
if (N = leaf node)
  generate word w with probability $P(w|C)$
else
  generate head daughter H with probability $P(H|C,exp)$
  where $exp \in \{unary, left, right\}$
  if (N = binary node)
    generate non-head daughter S with probability $P(S|C,exp,H)$
    where $exp \in \{left, right\}$

Figure 3.1: The Baseline generative CCG model (adapted from Hockenmaier, 2003)

	Expansion	HeadCat	NonHeadCat	LexCat		Head word	
Model	$P(exp\|\ldots)$	$P(H\|\ldots)$	$P(S\|\ldots)$	$P(c_S\|\ldots)$	$P(C_{TOP}\|\ldots)$	$P(w_S\|\ldots)$	$P(w_{TOP}\|\ldots)$
Baseline	P	P,exp	P,exp,H	-	-	-	-
+ $Conj$	$P,conj_P$	$P,exp,conj_P$	$P,exp,H,conj_P$	-	-	-	-
+ Grandparent	P,GP	P,GP,exp	P,GP,exp,H	-	-	-	-
LexCat	P,c_P	P,exp,c_P	$P,exp,H\#c_P$	$S\#H,exp,P$	$P=TOP$	-	-
LexCatDep	P,c_P	P,exp,c_P	$P,exp,H\#c_P$	$S\#H,exp,P\#c_P$	$P=TOP$	-	-
HeadWord	$P,c_P\#w_P$	$P,exp,c_P\#w_P$	$P,exp,H\#c_P\#w_P$	$S\#H,exp,P$	$P=TOP$	c_S	c_P
HWDep	$P,c_P\#w_P$	$P,exp,c_P\#w_P$	$P,exp,H\#c_P\#w_P$	$S\#H,exp,P$	$P=TOP$	$c_S\#P,H,S,w_P$	c_P

Table 3.1: Structure of generative CCG models (Hockenmaier, 2003): The # sign indicates a *back-off* level, e.g. $A,B\#C,D$ means that we interpolate $\hat{P}(\ldots|A,B,C,D)$ with $\tilde{P}(\ldots|A,B,C)$ which is an interpolation of $\hat{P}(\ldots|A,B,C)$ with $\tilde{P}(\ldots|A,B)$.

performs rather well (first row in table 3.2); a fact that can be attributed to CCG's ability to encode extended syntactical information –like subcategorisation frames– in its categories. Hockenmaier goes on to describe several extensions to this baseline model; we will examine the most important of them based on their overall performance.

The first two extensions add non-lexical information to the baseline model. These are the *Coordination* ($Conj$) and *Grandparent* features. $Conj$ is a binary feature that is true for constituents which expand to coordination on the head path. It is used as a conditional variable in the model as it is shown in table 3.1. This feature is used to ensure that, for a sentence without traces or right node raising, a CCG derivation with a type-raised subject composed with the verb is very unlikely.

The Grandparent feature is essentially the addition of the label of the parent node to the label of each node. This feature has shown to have a substantial effect on the

Model	NoParse	LP	LR	⟨⟩
Baseline	6	72.8	72.4	84.3
+ *Conj*	9	73.8	73.9	85.1
+ Grandparent	91	77.1	77.6	87.9
LexCat	9	75.8	76.0	86.8
LexCatDep	9	75.7	75.9	86.6
HeadWord	8	77.9	78.0	88.3
HWDep	8	81.6	81.9	90.1
HWDep(+ tagger)	7	81.4	85.6	89.9

Table 3.2: Performance for generative CCG models – Treebank section 23 (Hockenmaier, 2003)

performance of PCFGs (LP/LR: from 73.5%/69.7% to 80.0%/79.2%; Johnson, 1998) but the improvement is not that significant in case of CCG (third row on table 3.2). Moreover the lexical coverage of the parser is reduced leading to a large number of failed parses (91 compared to the second worse figure of 9). As Hockenmaier (2003) reports, both results are to be expected; CCG categories already encode information about parents and grandparents and therefore a history feature has less overall impact. In addition, since CCG's category set is quite large, appending the parent label leads to an even more fine-grained set which results in data sparseness.

In the last set of models Hockenmaier (2003) adds lexical information on which the probability distributions are conditioned. More specifically the expansion of each constituent is now conditioned on the lexical category c_P and the head word w_P of the constituent generated at their maximum projections (i.e. either at the root of the tree or when generating a non-head daughter).

As we can see from table 3.2, the performance of the parser is significantly increased by adding lexical information and even more when lexical category or word-word dependencies are introduced (models LexCatDep and HWDep respectively). It is clear that the best model is HWDep which captures 90.1% of the word-word dependencies in the test corpus (section 23 of the Penn Treebank). However since all of the aforementioned experiments use an already-tagged test corpus it was assumed that the input was of higher quality than can be achieved by a using POS-tagger. Therefore, an experiment was performed where the test corpus was tagged with the POS-tagger of Ratnaparkhi (1996). The results (shown in the last row of table 3.2) are slightly worse,

which is to be expected since the parser relies heavily on the POS-tags for unknown words. Nevertheless the final result of 89.9% (89.7% on section 00) is very close to that of Collins' best model[8].

3.1.2.2 Discriminative models

The next wide-coverage CCG parser was presented by Clark et al. (2002) and uses a discriminative model based on Collins (1996). This approach is quite different than the generative models discussed earlier in that it tries to calculate the probabilities directly. Another major difference is that Clark et al. use the notion of *dependency structures* instead of parse trees for their calculations. A dependency is defined as a 4-tuple $\langle h_f, f, s, h_a \rangle$, where h_f is the head word of the functor category, f is the functor category, s is the argument slot and h_a is the head word of the argument.

We start with maximum likelihood probability presented in chapter 2 –redefined in equation 3.4 with the sentence string α replaced by a sequence S of word, POS-tag pairs $S = \langle w_1, t_1 \rangle, \langle w_2, t_2 \rangle, \ldots, \langle w_n, t_n \rangle$.

$$\tau^* = arg\max_{\tau} P(\tau|S) \tag{3.4}$$

We follow Clark et al. (2002) and assume the best parse derivation is equal to the best dependency structure π which is a $\langle C, D \rangle$ pair where $C = c_1, c_2, \ldots, c_n$ is the sequence of categories assigned to the words $w_1, w_2, \ldots, w_n$ and $D = \{\langle h_{f_i}, f_i, s_i, h_{a_i} \rangle | i = 1, \ldots, m\}$ is the set of dependencies.

$$P(\pi|S) = P(C, D|S) = P(C|S)P(D|C, S) \tag{3.5}$$

It is now a question of calculating the two probabilities of the product: the probability of the category sequence given the sentence $P(C|S)$ and the probability of the set of dependencies given the categories and the sentence $P(D|C, S)$. The first of them can be approximated as follows:

$$P(C|S) \approx \prod_{i=1}^{n} P(c_i|X_i) \tag{3.6}$$

where X_i is the local context for the ith word. This probability can now be calculated using a *supertagger*[9] –a tool that extracts the local context of a word and uses maximum entropy techniques (following Ratnaparkhi (1996)) to calculate $P(c_i)$.

[8] In Collins (1997) only the standard PARSEVAL results are presented. Hockenmaier (2003) compares the word-word dependency results with those reported in Collins (1999) –90.9% of word-word dependencies recovered.

[9] The supertagger for this parser is described in Clark (2002). Since the parser of this project is also equipped with a supertagger, we will examine this tool further in chapter 4.

To calculate the dependency probabilities Clark et al. (2002) assume that each argument slot is filled independently:

$$P(D|C,S) = \prod_{i=1}^{m} P(h_{a_i}|C,S) \tag{3.7}$$

where h_{a_i} is the head word filling the argument slot of the ith dependency and m is the number of dependencies in C.

Finally the estimation of $P(h_{a_i}|C,S)$ is based on the following intuition: "given a pair of word, with a pair of categories, which are in the same sentence, what is the probability that the words are in a particular dependency relationship?". To help with this estimation, Clark et al. (2002) also define a number of auxiliary functions (where a,c are words and b,d are lexical categories):

- $C(\langle a,b\rangle,\langle c,d\rangle)$ is the number of times that the word-category pairs $\langle a,b\rangle$ and $\langle c,d\rangle$ are in the same word-category sequence in the training data.
- $C(R,\langle a,b\rangle,\langle c,d\rangle)$ is the number of times that $\langle a,b\rangle$ and $\langle c,d\rangle$ are in the same word-category sequence, with a and c belonging in dependency relation R.
- $F(R|\langle a,b\rangle,\langle c,d\rangle)$ is the probability that a and c are in dependency relation R, given that $\langle a,b\rangle$ and $\langle c,d\rangle$ are in the same word-category sequence.

The probability $P(h_{a_i}|C,S)$ is now approximated by:

$$P(h_{a_i}|C,S) \approx \frac{\hat{F}(R|\langle h_{f_i},f_i\rangle,\langle h_{a_i},c_{a_i}\rangle)}{\sum_{j=1}^{n}\hat{F}(R|\langle h_{f_i},f_i\rangle,\langle w_j,c_j\rangle)} \tag{3.8}$$

where $\hat{F}$ is the relative frequency estimator of F, calculated as follows:

$$\hat{F}(R|\langle a,b\rangle,\langle c,d\rangle) = \frac{C(R,\langle a,b\rangle,\langle c,d\rangle)}{C(\langle a,b\rangle,\langle c,d\rangle)} \tag{3.9}$$

An extension to this model is presented in Clark and Curran (2004b) where they approach the dependency structure probabilities using *Log-Linear* (or Maximum Entropy) models.

The parse probability of equation 3.4 is redefined in the log-linear model as follows:

$$P(\tau|S) = \frac{1}{Z_S} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\tau)} \tag{3.10}$$

where $\boldsymbol{\lambda}.\boldsymbol{f}(\tau) = \sum_i \lambda_i f_i(\tau)$ and f_i is a *feature* of the parse that can be any real-valued function with an associated *weight* λ_i. Z_S is a normalisation constant that ensures that

	LP	LR
Dependency Model	86.7	85.6
Normal-form Model	86.4	86.2

Table 3.3: Performance of Log-Linear CCG models – Treebank section 00 (source: Clark and Curran (2004b))

	LP	LR
Clark et al. (2002)	81.9	81.8
Hockenmaier (2003)	84.3	84.6
Clark and Curran (2004b)	86.6	86.3

Table 3.4: Comparison of wide-coverage CCG parsers (best models only) – Treebank section 23 (source: Clark and Curran (2004b))

$P(\tau|S)$ is a probability distribution (i.e. $P(\tau|S) \in [0,1]$) and is calculated as:

$$Z_S = \sum_{\tau' \in \rho(S)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\tau')} \tag{3.11}$$

where $\rho(S)$ is the set of possible parses for the sentence S.

The best parse is estimated by maximising the conditional (log)likelihood of the model given the data. The details for this estimation are quite technical and certainly beyond the scope of this review.

Clark and Curran also define a normal form model, comparable to the dependency model of Hockenmaier (2003). The results for both models are presented in table 3.3. Finally, a comparison of all the statistical CCG models is presented in table 3.4. Note that in Hockenmaier and Steedman (2002) the parser did not produce the predicate-argument structure that both Clark et al. (2002) and Clark and Curran (2004b) did and had to use their parser to achieve comparable results.

3.2 Natural Logic

As we have already mentioned, since the beginning of the history of logic there has been a search to construct a *Natural Logic* that would permit the description of the most important inferences of natural language using the terms of syntactic forms. While the generative semanticists of the 70s called for a *unique* Natural Logic, capable of

expressing "all and only the linguistically expressible valid inferences" (Lakoff, 1970), one could still discover interesting deductive systems which would approximate the most common linguistic inferences, without accepting such a strong assumption.

Both Sánchez-Valencia (1991) and van Benthem (1991) have underlined the significance of directional monotone inference patterns in natural language. In his Ph.D. thesis Sánchez-Valencia (1991) (and subsequently in Sánchez-Valencia, 1995) studies these phenomena and proposes a *monotonicity calculus*, a proof system for a fragment of Natural Logic using Lambek Grammar (a form of categorial grammar). He begins by defining upward and downward monotonicity (see section 2.2) in terms of a partial ordering among the denotations of the various logical types in Lambek Grammar.

Sánchez-Valencia (1991) then proceeds to insert monotonicity marks in the syntactic categories of the grammar thus introducing new logical types for functor expressions which can apply an upward (or downward) monotonicity to their arguments. For example in the case of lexical determiners, the new lexical categories would be as follows[10]:

(3.12) **a(n)**: $((e,t)^+,((e,t)^+,t))$
every: $((e,t)^-,((e,t)^+,t))$
no: $((e,t)^-,((e,t)^-,t))$

This *lexical monotonicity marking* is the first step in Sánchez-Valencia's system. But as Dowty (1994) points out, such monotonicity assignments are not sufficient to predict every inference in natural language. For instance the above type for **a(n)** would correctly predict an upward monotone entailment for both of its arguments in a positive polarity environment (e.g. the sentence *A dog walks* can entail both *An animal walks* and *A dog moves*). However we wouldn't get an upward entailment from the noun **dog** in *John doesn't own a dog* despite the initial lexical marking.

To deal with this problem and to correctly predict the final polarity of each constituent Sánchez-Valencia employs two more steps. The first performs an (external) monotonicity marking which adds $+$ and $-$ marks to each node in a derivation tree using the following rules:

[10]Since the reader may be unfamiliar with the notation of categorial grammar we must point out that in functor category (α,β) the arguments are represented by the term α. Also, note that this type of categorial grammar uses only logical types as categories where (e,t) is a mapping from entities e to truth values t.

(3.13) a.
$$\frac{(\alpha,\beta) \qquad \alpha}{\beta} \Longrightarrow \frac{\begin{array}{cc}(\alpha,\beta) & \alpha\\ + & \end{array}}{\beta}$$

a.
$$\frac{(\alpha^{+},\beta) \qquad \alpha}{\beta} \Longrightarrow \frac{\begin{array}{cc}(\alpha^{+},\beta) & \alpha\\ + & +\end{array}}{\beta}$$

a.
$$\frac{(\alpha^{-},\beta) \qquad \alpha}{\beta} \Longrightarrow \frac{\begin{array}{cc}(\alpha^{-},\beta) & \alpha\\ + & -\end{array}}{\beta}$$

These rules, when applied to all of the derivations, produce a monotonicity-marked path from each node to the root of the sentence.

In the third step called *polarity determination* we traverse these paths and assign the actual direction of inference to each constituent. This is accomplished by these rules:

(3.14) If D is a syntactic derivation with root α, then:

a. A node γ has polarity in D iff there is an uninterrupted chain of polarity symbols in the path from γ to α.

b. A node γ is *positive* in D iff γ has a polarity in D and the number of negative marked nodes in this path is even.

c. A node γ is *negative* in D iff γ has a polarity in D and the number of negative marked nodes in this path is odd.

To get a better grasp of these three steps, figure 3.2 presents an example that will provide intuitive information about the process.

While Sánchez-Valencia's system is both sound and complete it has a number of shortcomings. First, it involves three separate steps, which results in a rather indirect linking between logic and the natural language expressions that licence the changes in monotonicity (Dowty, 1994). Another problem is that the $+$ and $-$ symbols have different connotations when used in a monotonicity-marked tree and in a polarity summary. For instance a $+$ marked constituent can have a $-$ polarity and vice versa as we can see in figure 3.2.

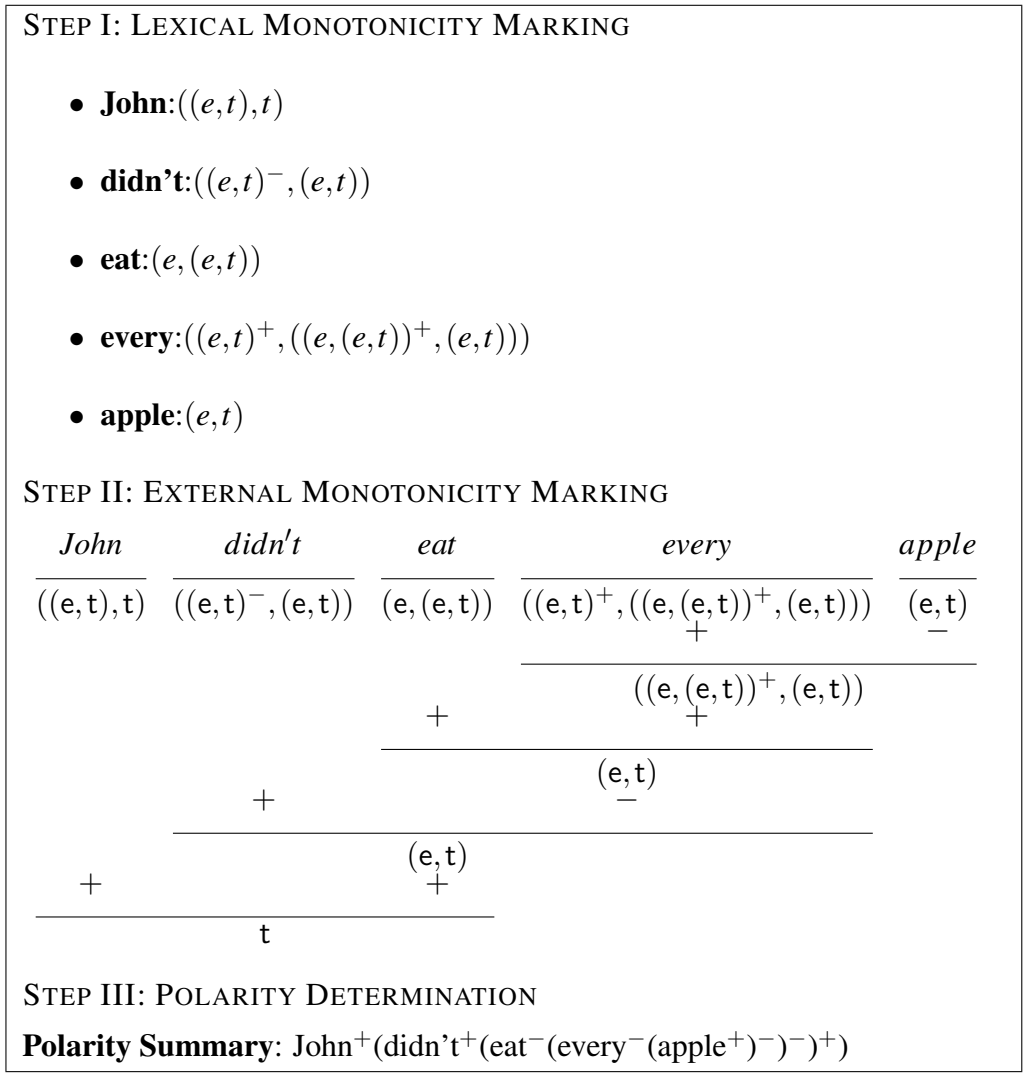


Figure 3.2: Example of polarity determination in Sánchez-Valencia (1991)

Furthermore Sánchez-Valencia (1991) doesn't account for non-monotonic phenomena like coordination and relative clauses, as he only considers a small fragment of the English grammar[11].

These problems were addressed in subsequent extensions of Sánchez-Valencia's work. Dowty (1994) presents an alternative formulation of Sánchez-Valencia's calculus that he uses to describe negative polarity-licensing expressions.

Dowty collapses the steps of Monotonicity Marking and Polarity Determination into the syntactic derivation by allowing the constituents to be generated with the markings that they would receive at the final step, already in place. He also introduces the use of two new polarity types namely *polarity preserving* and *polarity reversing* to address the phenomenon of one constituent appearing with a positive polarity in one derivation and a negative polarity in another.

Finally, Dowty also examines the relationship between *Negative Polarity Items* (NPIs) and downward monotonicity licensing. In particular he identifies that every NPI may be described as a polarity-reversing category that in a positive sentential environment, licenses downward entailments.

Sánchez-Valencia (1991) uses order statements (like $[\![\textbf{tall man}]\!] \leq [\![\textbf{man}]\!]$) to construct the natural logic entailments but provides no formal proof for them. Fyodorov et al. (2003) extend Sánchez-Valencia's original monotonicity calculus to include a formal *order calculus* for defining order statements as purely syntactic relations between derivation trees.

The proof system of Fyodorov et al. also provides treatment for coordination, allowing a simple account of non-monotonic expressions that can be reduced to conjunctions of monotonic ones. They finally describe an initial implementation of their proof system for computing natural logic inferences but provide no results of that effort.

The first computational implementation of a natural logic system (to our knowledge) is provided by van Eijck (2005). Van Eijck describes a new monotonicity marking algorithm that, in contrast to all existing monotonicity calculi (which follow Sánchez-Valencia's work), applies the marking rules in a top-down approach. Like Dowty (1994), he uses the preservation and reversal polarity markings. The monotonicity marking algorithm is described in figure 3.3.

The algorithm was implemented into a Haskell natural logic inference that provides very satisfying results, but only for a small fragment of English grammar.

[11] This is a common practice among all the extension and related work in Natural Logic. The only work to date that studies a full-fledged portion of English is by MacCartney and Manning (2007), which is examined later on.

Root Marking: The main structure C to be marked has positive polarity.
Component Marking: If a structure C has polarity mark m then:
Leaf Marking: If C is a leaf, then done.
Composite Marking: If C consists of a function F and argument A, then F has polarity marking m, and A has polarity marking $f(m)$ where f is the polarity marking map that encodes the monotonicity behaviour of F.

Figure 3.3: Top-down monotonicity marking (source: van Eijck, 2005)

3.2.1 Natural Logic in Textual Entailment

The first use of a Natural Logic system in a real-world application was made by MacCartney and Manning (2007) in the field of Textual Entailment Recognition. According to Dagan et al. (2006), *Textual Entailment* is defined as "a directional relationship between pairs of text expressions, denoted by T –the entailing *Text*, and H –the entailed *Hypothesis*". We say that T entails H if the meaning of H can be inferred from the meaning of T, as would typically be interpreted by an average language user.

There are usually two approaches used in textual entailment recognition. The first is called "shallow" inference and refers to the use of statistical methods to identify a syntactic alignment between the constituents of the text and the hypothesis. The closer the matching between the constituent, the more probable it is for the hypothesis to be entailed by the text. Even though this may seem a "naive" approach, these methods coupled with lexical relation and word similarity measures, provide the best results to date (Giampiccolo et al., 2007).

The other approach, called "deep" inference, involves the use of formal logic combined with theorem proving systems to produce a logical proof of the entailment. Systems like that of Bos and Markert (2006) provide an impressive precision (>75%) but have significant problems in terms of coverage (achieving less than 5%).

MacCartney and Manning (2007) suggest Natural Logic inference as an intermediate stage of "logical intensity". They describe the implementation of a wide-coverage system based on the three-step monotonicity calculus of Sánchez-Valencia (1991). By using only *substitution*, *deletion* or *insertion* edits[12] their NatLog system can incrementally process the hypothesis and the text to achieve an optimal alignment of their

[12]These edits relate directly to the direction of monotonicity. Thus a constituent can be substituted with a broader or more specific one; it can be deleted from the text or inserted in the hypothesis.

constituents. The edits concern primarily noun phrases or single determiners. The following example shows how a text and a hypothesis can be aligned using the NatLog system (the ADV edit simply advances to the next constituent):

(3.15) Text: An Irishman won a Nobel Prize.
Hypothesis: An Irishman won the Nobel Prize for literature.

An Irishman	$\Longrightarrow$	An Irishman	ADV
won	$\Longrightarrow$	won	ADV
a	$\Longrightarrow$	the	SUB
Nobel prize	$\Longrightarrow$	Nobel prize	ADV
	$\Longrightarrow$	for literature	INS
.	$\Longrightarrow$	.	ADV

While by itself the system achieves 76% precision (similar to that of Bos and Markert, 2006), the coverage is not significantly improved ($\sim$ 24%). MacCartney and Manning also combined the NatLog system with a conventional textual entailment system where it provided a statistically significant accuracy gain of 3.12%. However the final performance of the hybrid system (Accuracy/Precision: 63.62%/63.74%) is still not at state-of-the-art levels (Giampiccolo et al., 2007).

3.3 Word Sense Disambiguation

As we have mentioned in section 2.3, the quest for sense disambiguation began in the late 40s. It was in fact Warren Weaver in his famous mimeographed memorandum (Weaver, 1955) that recognised the problem of a word having multiple meanings[13] as one major challenge in Machine Translation and proposes the first theoretical approach to its solution:

> If one examines the words in a book, one at a time as through an opaque mask with a hole in it one word wide, then it is obviously impossible to determine, one at a time, the meaning of the words.[...]
> But if one lengthens the slit in the opaque mask, until one can see not only the central word in question but also say N words on either side, then if N is large enough one can unambiguously decide the meaning of the central word.[...]

[13] However, in the main body of his study he describes the "easier" case of a word having only one meaning in a given text of a specific domain (e.g. Mathematics).

> The practical question is: "What minimum value of *N* will, at least in a tolerable fraction of cases, lead to the correct choice of meaning for the central word?"

Since then, and especially during the 70s, a large number of approaches have been suggested, mainly inspired by the research in psycholinguistics and connectionist theories of language. Although theoretically interesting and well motivated, these approaches were of little (if any) practical use in real-world WSD applications. Their common weak point was the need for enormous amounts of hand-crafted knowledge[14].

The solution for this "knowledge acquisition bottleneck" (Gale et al., 1992) came in the mid 80s when large-scale electronic lexical resources (dictionaries, thesauri and various corpora) became available[15]. This shift also signaled the transition from linguistically motivated theories to empirical methods of NLP.

3.3.1 Thesauri and Machine-Readable Dictionaries

The first knowledge-based attempt to the problem of WSD was made by Masterman (1957) where she used a knowledge base constructed by *Roget's Thesaurus* for disambiguating word senses in machine translations. The project had limited success and the project was abandoned; several years later, researchers at the University of Kansas used the thesaurus approach again this time to disambiguate between verbs (Ide and Véronis, 1998).

The biggest advancement in the use of thesauri for WSD came with the work of David Yarowsky (Yarowsky, 1992). In a hybrid knowledge and corpus based approach he extracted a 100-word context from a corpus (the 1991 electronic version of *Grolier's Encyclopedia*) for each word in *Roget's* most common categories and used statistical analysis to determine the words that are most likely to co-occur with the category words, thus forming sense classes. Then, he used Bayes' Rule to determine to which of these classes each ambiguous word is more likely to belong. He states that the accuracy of the system is approximately 92% on 12 ambiguous words with a mean of 3 senses per word.

Since thesauri represent only a crude approximation of lexical knowledge, most of

[14] We will focus here on knowledge-based approaches as they are the most relevant to this project. For a comprehensive introduction to early AI-based methods see Ide and Véronis (1998). For a review of the current WSD approaches the book by Agirre and Edmonds (2007b) is an excellent resource.

[15] This is not entirely true as the first freely available electronic dictionaries became available in 1968 and the first electronic thesaurus even earlier in 1957 (Ide and Véronis, 1998); their use in NLP however was not established until the 80s.

the knowledge-based WSD research turned early on to *Machine-Readable Dictionaries* (MRDs). Michael Lesk was one of the first researchers to create a WSD algorithm using a MRD to represent lexical knowledge about the word senses. In his rather short paper (Lesk, 1986) he proposes "a cheap solution to the problem of sense discrimination" based only on the overlaps of dictionary definitions of the various senses. This method is reported to have achieved 50-70% correct disambiguation using a typical MRD (Ide and Véronis, 1998). Since it first appeared, it has been used in numerous variations by many systems and in way or another every knowledge-based system uses it. For a review of some of these variations see Mihalcea (2007).

Several enhancements of the Lesk method have been proposed; Véronis and Ide (1990) suggest the use of Lesk's algorithm in a neural network created from MRD definitions. In particular, each node (representing a word) is linked to its sense, which are in turn linked to words in their definitions and so on. Using various adjustments they report success rates between 72 and 85% on experiments with 23 ambiguous words (Ide and Véronis, 1998).

Another extension was the addition of supplementary fields of the dictionary definitions such as *box codes* and *subject codes* from the *Longman Dictionary of Contemporary English* (Krovetz and Croft, 1989; Guthrie et al., 1991). These encode semantic information about the words, including semantic primitives (ABSTRACT, ANIMATE, HUMAN, etc.) and type restrictions in the case of box codes as well as subject primitives (ECONOMICS, ENGINEERING, etc.) in the case of subject codes. However, no quantitative results can be found for these methods. Proper (quantitative) evaluation was a major issue for all of these approaches during the 80s-90s period.

3.3.2 Computational Lexicons and Large-scale Knowledge Bases

In parallel to MDRs and electronic thesauri, the creation of large-scale hand-crafted knowledge bases started around the mid 80s. Massive amounts of knowledge, lexical and ontological, were encoded into computational lexicons such as *WordNet* (Fellbaum, 1998) and knowledge bases such as *Cyc* (Lenat, 1990).

Their use as WSD tools didn't take long to arrive and, in the case of WordNet, its large coverage combined with its availability make it the most widely used lexical resource for disambiguation.

Most, if not all of the work in this field is based on the concept of *semantic similarity*, where a set of documents or terms within term lists are assigned a metric based

on the likeness of their semantic content.

One of the most influential contributions was that of Philip Resnik. His system, based on earlier work with WordNet (Sussna, 1993) uses the *shared information content* of the words to calculate the specificity of the sense that subsumes two words (only in the IS-A hierarchy of WordNet). In this way the less distance he has to traverse in order to find a specific concept that subsumes two words, the more related these words are; in other words the more specific the root concept is, the more related the words it subsumes (Resnik, 1995). For example, the *Medical Doctor (M.D.)* sense of the word **doctor** is more related to the meaning of **nurse** than the *PhD graduate* sense; the most specific concept that subsumes both meanings in the first case is $< health\ professional >$ whereas in the latter is $< person, individual >$ and therefore less specific.

A rather similar approach, this time using the Cyc ontology, is taken by Curtis et al. (2006). However, instead of using only taxonomic properties of the concepts (like the IS-A hierarchy of WordNet), they exploit a rich higher-order array of properties in order to calculate the semantic similarity of two words. In addition they explore the *semantic contribution* that each word makes to the content of the target text. This means that the system creates an explicit model of the document structure and favors word senses that contribute to a consistent semantic interpretation of the model[16]. For instance, the word **book** is lexically ambiguous between the sense of *authored work* and *book copy*; however when the system parses the phrase "worn book" only the interpretation of the physical object (*book copy*) can be kept because, unlike a conceptual work, it can undergo wear and tear. The researchers report markedly better results than chance (more than 60% improvement) but still these are not comparable to the results from more statistically oriented approaches[17].

[16]These interpretations are made in a higher-order logic based on predicate calculus.

[17]However, comparisons are very difficult to draw here, as most of these studies use different sets of ambiguous words and different test corpora to produce their results.

Chapter 4

Methodology

4.1 Parsing

As we saw in section 3.1.2, there are at least two implementations of wide-coverage CCG parsers available (Hockenmaier, 2003; Clark and Curran, 2004b). In principle either of them could have served as the basis for this project. However there are several reasons that lead us to the decision to implement our own statistical CCG parser.

To understand the most important reason for this decision, we must stress again that the aim of the project is to create a platform for natural logic based inference, not a parsing system; our efforts are mainly concentrated on the optimisation of the monotonicity environment and the entailment relationships between the predicates of the sentence.

This means that we must modify the behaviour of the unification rules, add monotonicity information to the syntactic categories (and their semantic representations) and even alter some aspects of the training corpus. Some of these modifications would have been very difficult, if not impossible, to achieve without rewriting a very large part of the code of the parsers.

Another reason for not using a purely statistical parser is the way both parsers implement the application of the CCG rules. They choose to unify two constituents based only on whether the specific unification has a probability mass, not on whether there is a CCG rule that will allow this unification. Apart from the fact that this approach is theoretically inconsistent with pure CCG, it raises a series of problems for the monotonicity marking process. We will examine such a problem in section 4.1.5.

Furthermore, the implementation of a statistical parser can be justified by its educational value. The implementation process provided access to invaluable information

about the theory and practice of parsing which would not be available had we chosen to use an off-the-shelf parser.

Finally, we have implemented the parser in a modular way, so that every component can be easily replaced; the highly portable Java language was chosen for the same reason. This provided us with a highly customisable parsing tool that can be used for other research projects (cf. Thomforde, 2008).

4.1.1 OpenCCG Platform

The above mentioned restrictions do not necessarily imply that we had to implement a wide-coverage parser from scratch. Instead we chose to build a "wide-coverage" system on top of an existing CCG parser called *OpenCCG*. OpenCCG is "an open source natural language processing library written in Java, which provides parsing and realization services based on the CCG formalism" (White, 2008). It was developed by Jason Baldridge as a successor to the *Grok* system, which was an implementation of Baldridge's multi-modal extension to CCG (discussed in section 2.1.1).

OpenCCG was designed to support all of CCG's features, namely slash modalities, semantic interpretations and even intonational features. It also supports various syntactic and morphological features (e.g. case, gender and number information). Using these structures one can construct a CCG grammar by creating a lexicon (i.e. a set of lexical entries using any of these features as extra information) and by specifying which CCG rules are used by the grammar. In this way, OpenCCG provides a truly "open" environment that can support the creation of a grammar for any language, not only English. Indeed OpenCCG has been used in a variety of projects in many languages both as a parser and a realiser (Baldridge, 2008).

The parsing process in OpenCCG is based on the CKY algorithm described in section 2.1.2. However, since this was originally a non-statistical parser, it uses a different approach for the unification process compared to the CCG parsers of Hockenmaier (2003) and Clark and Curran (2004b). As we see in figure 4.1 OpenCCG relies on the CCG rules (both unary and binary) defined for the specific grammar. This is in contrast to the way its statistical counterparts are implementing the unification. Most statistical parsers (the two wide-coverage CCG parsers included) are using a top-down tree expansion approach for the creation of each constituent. This means that a statistical parser will unify two constituents as long as there is at least one example of this unification in the training data. For instance, to use one example from the CCGbank,

```
unify input:
  if (input.size = 1)
    for (rule_x ∈ UnaryRules)
      derivation = rule_x.result
  else if (input.size = 2)
    for (rule_x ∈ BinaryRules)
      case (rule_x instanceof ForwardRule)
        if (unifies_{rule_x} (input_1, input_2))
          derivation = input_1.target
      case (rule_x instanceof BackwardRule)
        if (unifies_{rule_x} (input_2, input_1))
          derivation = input_2.target
return derivation
```

Figure 4.1: The unification algorithm of OpenCCG

the "." constituent is allowed to unify with the derived "S" structure to its left even though there is no CCG rule that would allow a unification of this type[1].

OpenCCG also features a chart unpacking mechanism designed to deal with the problem of spurious ambiguity (see section 2.1.1). Eisner (1996) has proposed an elegant solution for this problem –essentially forcing CCG to produce normal form parses– which has been used in Hockenmaier (2003). OpenCCG uses a similar idea to Eisner (1996)'s, which allows for semantically equivalent parses (stored as chart edges) to be "packed" as alternatives to a single representative edge which is then used throughout the parsing process.

Using OpenCCG as a platform, we proceeded to implement a statistical framework that will provide us with the final wide-coverage parsing tool for our inference system. This process involved several steps which we shall now examine.

[1]Even though this example uses punctuation which could be regarded as an exceptional case, CCGbank contains derivations among standard constituents that could not have been produced by any CCG rule. One example is the **n't** particle which is discussed in section 4.1.5.

4.1.2 Suppertagging

We start from the most basic element of a CCG parser; the wide-coverage lexicon. All non-statistical parsers come with a small, hand-built lexicon that enables only a fraction of the language to be parsed, and OpenCCG is no exception. One way to obtain a wide-coverage lexicon is to collect all the lexical entries from a manually annotated corpus such as CCGbank. We can then assign to each word of the unseen sentence all the categories from the word's entry in the lexicon. This approach has been used by Hockenmaier (2003).

An alternative is to use a statistical method of assigning categories to words. This way, a statistical model will assign the best (most probable) categories to each word according to its sentential context. This approach, called *supertagging*, was first introduced for Lexicalised Tree Adjoining Grammars by Bangalore and Joshi (1999) in order to reduce the number of structures assigned to each word and thus increase the parse accuracy. It was applied to CCG by Clark (2002) and its efficiency was shown in Clark and Curran (2004a) to increase significantly the speed of the parser by reducing the derivation space[2].

This increase in speed, as well as the reduction in memory requirements, is vital to our implementation, as OpenCCG is written in Java. In order for Java to be highly portable it relies on the creation of a Virtual Machine –a collection of software programs that provide an interface between the high-level Java code and the machine instructions. This allows the Java code to be independent of the hardware or the operating system of the user (unlike platform-dependent languages (e.g. C) where the user has to compile and build a version of the program specifically for his/her machine), but at the same time it adds one extra layer of computational processes. This results in slower performance and larger resource requirements especially in large applications.

The supertagger uses a log-linear model similar to the parsing models described in section 3.1.2.2. More specifically it calculates the probability $P(C|x)$ (where C is the lexical category and x is the context) as:

$$P(C|x) = \frac{1}{Z(x)} e^{\sum_i \lambda_i f_i(C,x)} \tag{4.1}$$

where f_i is a feature function, λ_i is the corresponding weight to the function and $Z(x)$ is a normalisation constant. The context is as a 5-word window that surrounds the word and the features are defined for each word in the window and for the part-of-speech

[2]Clark and Curran (2004a) report that the parsing speed is increased by a factor of 77; however, unlike Bangalore and Joshi (1999) they report only a minor increase in parsing accuracy

tags of each word. We will not go into details about the how the weights are calculated, other than that the process is based on the Generalised Iterative Scaling (GIS) method using Gaussian Priors for smoothing (Clark, 2002).

The supertagger can be used to assign the single most probable category to each word, achieving an accuracy of $\sim 92\%$. According to Clark (2002) however, this figure is not enough for the supertagger to be integrated in a parser, as it leads to a significant decrease in coverage. This led to the idea of a *multi-tagger* that assigns more than one category to each word. The multi-tagger works by first assigning the most probable category using equation (4.1) and then assigns all those categories whose probability is within a factor β of the highest probability.

For words that were seen at least k times in the training data, the supertagger chooses only from the categories that appear in the word's entry in a *word tag dictionary*. The tag dictionary contains, for each word entry, all the categories seen with that word in the training data. For words that are seen fewer than k times, the supertagger assigns categories from a *POS tag dictionary* that contains the categories seen with each part-of-speech tag.

Table 4.1 gives a per-word accuracy for various values of β along with the average number of categories per word and the percent of sentences that contain only correct category assignments. Unless otherwise specified, a value of $k = 20$ was used. These results show the reduction in the average number of categories per word that can be achieved by the supertagger while still maintaining a significantly high accuracy. Clark and Curran (2004a) use the example of the word **is**, which has 45 categories in its word tag dictionary entry and the supertagger manages to assign only one category (correctly) for $\beta = 0.1$ and 3 for $\beta = 0.01$.

The implementation of the supertagger for our system was based on the work of Dennis Mehay from Ohio State University who designed a Java version of the Clark and Curran supertagger[3] and a GIS trainer.

Finally, we use the Stanford part-of-speech tagger to assign part-of-speech information used as context features by the supertagger. The Stanford tagger uses a log-linear (maximum entropy) model inspired by the parser of Ratnaparkhi (1996). The details of the tagger are described in Toutanova and Manning (2000) and Toutanova et al. (2003). Used on Treebank tokens (on which it was trained) the tagger achieves 97.24% accuracy (89.04% on unknown words).

[3]The code for the project can be found at `http://code.google.com/p/statopenccg/`

β	Cats/Word	Accuracy (%)	Sent. Accuracy (%)
0.1	1.4	97.0	62.6
0.075	1.5	97.4	65.9
0.05	1.7	97.8	70.2
0.01	2.9	98.5	78.4
$0.01_{k=100}$	3.5	98.9	83.6
0	21.9	99.1	84.8

Table 4.1: CCG supertagger accuracy – CCGbank section 00 (Clark and Curran, 2004a). The $\beta = 0$ specifies that no constraining factor was used. The supertagger selected all the categories in the word's entry.

4.1.3 Statistical Models

Once the supertagger has selected the categories for each word, the actual parsing process takes place and the role of the statistical model will be to identify the most probable derivation. However, at this stage the differences between OpenCCG and the wide-coverage parsers are crucial. As we saw in section 2.1.2 the wide-coverage models approach parsing from a top-down perspective, starting from the topmost node and assigning probabilities to every expansion until they reach the leaf nodes (the words). This provides a *tightness* to the model which means that no probability mass is assigned to ungrammatical parses.

OpenCCG, as we have already mentioned (figure 4.1), proceeds in a bottom-up way, creating derivations only if there is a corresponding CCG rule (allowed by the particular grammar). This approach is, at least in principle, incompatible with any statistical model we have examined. However, as we have already mentioned our goal is to create an environment for natural logic inference where the parser is used as the means to obtain the relationships between the sentence constituents and not as an end by itself. Thus, the statistical model must only provide an indication of the most probable parse, while its tightness is not a requirement. In other words, the system can assume that each of the input sentences will be grammatical[4].

For these reasons we have chosen to implement the most basic of the models examined earlier. We based our model on the Baseline model of Hockenmaier (2003), for

[4]We could even assume that each derivation has to be a declarative sentence (S[dcl]), forcing the parser to search for the most probable categories that would yield such a derivation. However, for now, we will not enforce such austere restrictions.

its simplicity and its relatively good performance. The probability distribution families used are repeated below for convenience.

- Lexical probability: $P(w|C)$
- Expansion probability: $P(exp), exp \in \{unary, left, right\}$
- Head daughter probability: $P(H|C, exp), exp \in \{unary, left, right\}$
- Non-head daughter probability: $P(S|C, exp, H), exp \in \{left, right\}$

The probability for each derivation is calculated using the same algorithm as Hockenmaier (2003), described in figure 3.1. The probability of the final derivation is the product of the probabilities of all the preceding derivations. The final results are reranked based on their probability score and the parser returns the best one.

If at any time the model assigns a 0 probability to any of the distributions (because it is not seen in the training data), the unification fails even if it is allowed by the CCG grammar. Both this and the reverse situation (where a unification has been assigned a probability mass but is not allowed by any CCG rule) are the main reasons for the less than optimal performance of the parser. We will examine these problems more extensively in section 5.

4.1.4 Coordination and Punctuation

OpenCCG in its original form was unable to parse coordination structures. However since we wanted to maintain its rule-based philosophy we chose not to rely on a programming solution (as both Hockenmaier (2003) and Clark and Curran (2004b) do); instead we implemented a version of the ternary coordination rule (**Φ**) of Steedman (2000)[5]. The implementation is described in figure 4.2, while examples (4.2) to (4.4) present the output of the parser in derivations containing coordination.

In particular, example 4.4 shows that this rule can coordinate non-standard constituents obtained through the use of functional composition (**B**).

[5]In Steedman (2000) the coordination rule was proposed instead of the more permissive conjunction category $(X\backslash X)/X$ as it would lead to the violation of the across the board (ATB) condition (Ross, 1967). In more recent work, however (e.g. Steedman and Baldridge, 2007) Mark Steedman proposed the abandonment of the coordination rule and its substitution with the "modalised" $(X\backslash_\star X)/_\star X$ category which will combine like types by using *only* application rules. In the current stage of this project the automatic assignment of modalities is not facilitated; therefore the use of the coordination rule is the only possibility.

chart

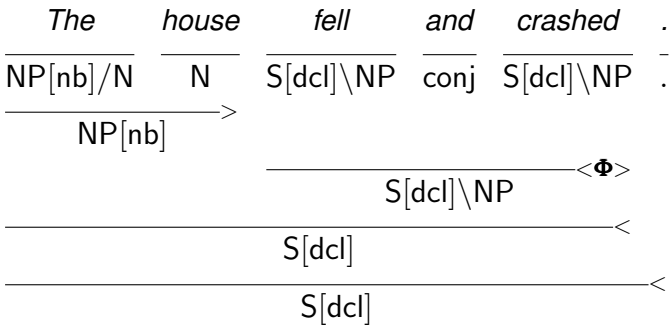
x

y	0	1	2	3	4
0	the(NP/N)	NP	S	0	[S]**
1	0	house(N)	0	0	0
2	0	0	fell(S\NP)	0	[S\NP]*
3	0	0	0	and(conj)	0
4	0	0	0	0	crashed(S\NP)

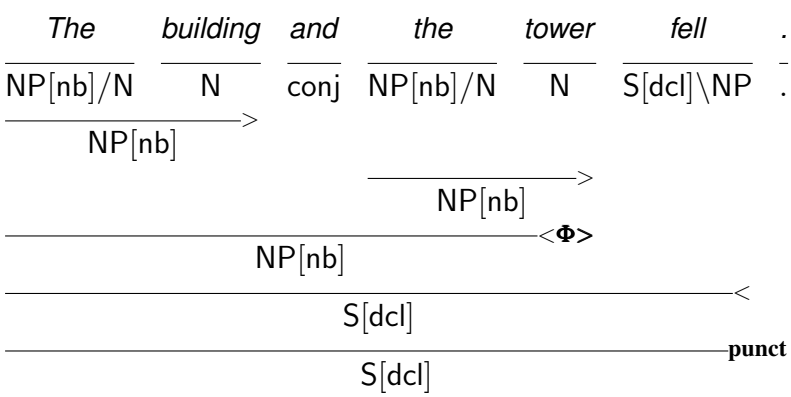
- Perform the original CKY algorithm
- **if** (no full parse):
  conj edge index: $conjInd = [x, y]$
  Neighbour constituent edge indeces:
  $firstN = [x-1, y-1]$, $secondN = [x+1, y+1]$
  for ($cur_Y = firstN(Y)$ to 0):
    for ($cur_x = firstN(x)$ to $\max(chart(y))$):
    if ($cur_x.sign$ == $cur_y.sign$):
      insert *derivedSign* at $chart[cur_x, cur_y]$ (*)
- Perform a "lightweight" version of CKY,
  (inserting cells only if they don't already exist (**))

Figure 4.2: The coordination rule algorithm

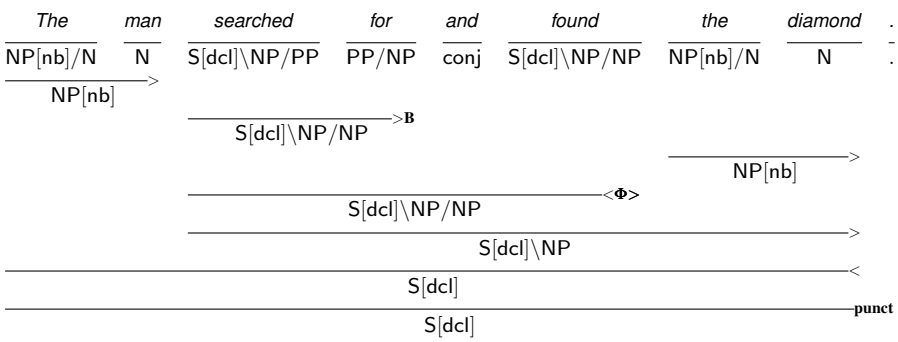
(4.2)



(4.3)



(4.4)



One other major component of a wide-coverage parsing missing from OpenCCG is the treatment of punctuation. However, unlike coordination which can be treated by a single rule, there is a variety of punctuation marks with different categories and functions within a sentence. For instance, while full stop (**.**) can be treated in most cases an $S\backslash S$, comma (**,**) may have a variety of different uses despite being marked with the same category. As we can see in figure 4.3, commas can be "behave" either as $NP\backslash NP$ (1st case) or as $(NP\backslash NP)/NP$ in the second example. Commas can also be used as coordinators in examples like this:

(4.5) John likes apples, bananas and oranges.

Every punctuation sign has similar uses, which for a truly statistical parser present a minor challenge, as it will only need relevant examples to exist in the training data.

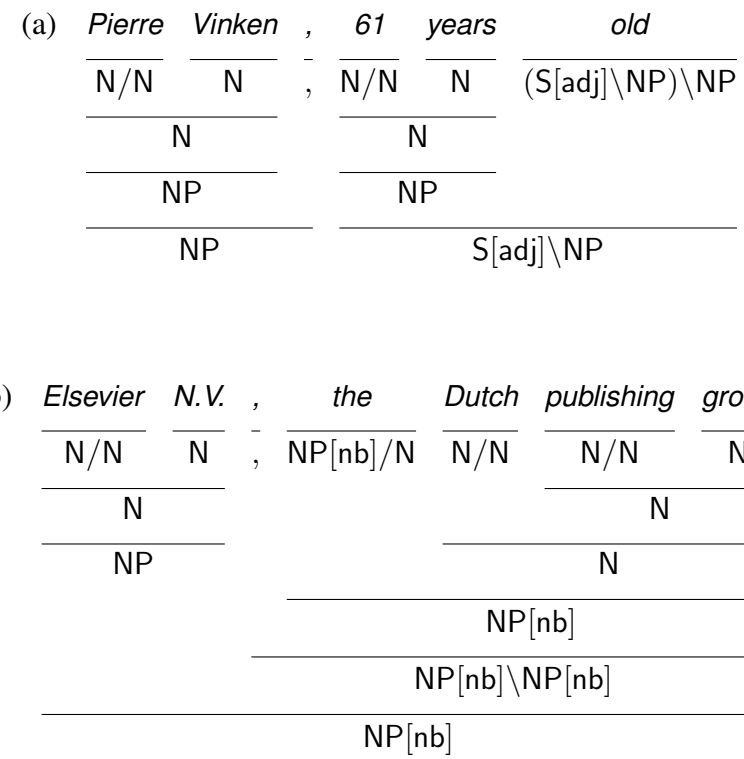


Figure 4.3: Punctuation treatment in wide-coverage CCG parsers

However this is a significant obstacle for our parser as there are no rules to allow the unification of a normal category with a punctuation category[6].

Because of the time limitations we have chosen to remove all punctuation except full stops, which were treated by a very simple *punctuation rule* that combines them to the previous most complete derivation. This is not very rare for application in the early steps of development (for instance, see Clark et al., 2002) although we had to make to sure that the punctuation was removed after the supertagger process, in order to preserve the contextual information.

4.1.5 Negation

Negation (expressed by the negative particle **not** and its contracted form **n't**) is not treated exceptionally in wide-coverage parsers, apart from the fact that (in the contracted form) it is treated as a separate lexical token which has to be combined with the preceding (auxiliary) verb. However, there are two reasons why we should treat negation separately in our system.

[6]There are some cases where the supertagger assigns the punctuation marks proper CCG categories along with the punctuation category. In this situation a full parse will be available without any extra rules. However, such cases are quite rare and even then a successful parse is not guaranteed.

The first reason has to do with the differences in the parsing procedure between OpenCCG and the pure statistical parsers discussed earlier. The fact that statistical parsers combine the various constituents based only on their appearance in the training data, makes the derivations extremely sensitive to annotation errors.

This was the case with the **n't** particle, whose most probable category is $(S\backslash NP)\backslash(S\backslash NP)$. The auxiliary verb **do** has $(S[dcl]\backslash NP)/(S[b]\backslash NP)$ as its most probable category. The result category of this unification has to be the same as the verb's in order to allow the parsing of the rest of the sentence. The only allowed way to achieve this derivation is by using the Backwards Crossing Composition rule repeated in 4.6b for convenience:

(4.6) CCG Crossing Composition Rules:

a. $X/Y \quad Y\backslash Z \quad \Rightarrow_{\mathbf{B}} \quad X\backslash Z \quad (>\mathbf{B}_{\times})$

b. $Y/Z \quad X\backslash Y \quad \Rightarrow_{\mathbf{B}} \quad X/Z \quad (<\mathbf{B}_{\times})$

This rule implies that **n't** should be the head daughter of the derivation and **do** the sister (non-head) constituent. Apart from being counter-intuitive, this assignment never occurs in the training data. Instead one can find numerous examples of the same derivation where **do** is the head constituent and **n't** the sister. However there is no rule that would allow such a derivation[7].

The second and perhaps more important reason is that negation is a very important concept for the natural logic inference and it is imperative that we treat it in the most uniform way. We have to ensure that the negative particles will have the same relative position in a derivation (either head or sister) in order for the monotonicity marking scheme (c.f. section 4.2) to work consistently.

It is for these reasons that we decided to create a new syntactic primitive called *neg* (similar to the conjunction (*conj*) and punctuation primitives) that we assigned to both negative particles by automatically pre-processing the training corpus. We have then implemented a new *negation rule* which is similar to the one used for the treatment of full stops: it simply attaches the negative particle to the previous constituent and returns its syntactic category as the result. Again here, we are making the assumption that all the input to the system will be grammatical sentences (where the negative particles can only follow an auxiliary verb).

[7] One might argue that Forward Crossing Composition (4.6a) could be used here; however the result category of the derivation in that case would be $(S[dcl]\backslash NP)\backslash(S[b]\backslash NP)$ which is not the same as the original category of **do**.

4.2 Polarity Marking

Having completed the syntactic analysis, we now turn to the core of natural logic inference, which is to determine the direction of monotonicity for each constituent. We will base our monotonicity marking method on the work of Sánchez-Valencia (1991) but we will apply a slightly different approach, similar to that of Dowty (1994). We will employ the external monotonicity marking and the definition of polarity steps in a single process that we shall call *polarity marking*.

In order to do this we added two more polarity marks (following Dowty, 1994) besides positive (X^+) and negative (X^-)[8]; X° which signifies that the constituent bears the same polarity value as its syntactic environment (i.e. *polarity preserving*) and $X^\bullet$ which means that the constituent bears the inverse polarity of its environment (i.e. *polarity inverting*). We can therefore assign polarity marks to each lexical category in our dictionary. Polarity marks for complex categories can be assigned to the target and each of the arguments (e.g. 4.7d).

(4.7) a. walks := $S^\circ \backslash NP$

b. owns := $(S^\circ \backslash NP)/NP$

c. gives := $((S^\circ \backslash NP)/NP)/NP$

d. some: = NP^+/N^+

A polarity mark assigned to an argument will be passed on to the sibling constituent in a derivation using the following conventions:

- $X^\pm\ X^\circ \rightarrow X^\pm$ (accordingly $X^\pm\ X^\bullet \rightarrow X^\pm$)
- $X^\circ\ X^\circ \rightarrow X^\circ$ (accordingly $X^\circ\ X^\bullet \rightarrow X^\bullet$)
- $X^\bullet\ X^\circ \rightarrow X^\bullet$
- $X^\bullet\ X^\bullet \rightarrow X^\circ$
- $X^\circ\ X^\pm \rightarrow X^\pm$ (accordingly $X^\bullet\ X^\pm \rightarrow X^\pm$)

Using the same notation we can add polarity marks to lexical constituents that could be directly affected by the changes in the environment cased by the derivations[9].

[8] Normally there are also constituents with unspecified polarity. However, since this is an automatic process of polarity assignment, every constituent will bear a polarity mark.

[9] Note that, to be closer to the semantic transparency that CCG can offer, we would have to employ the polarity marks on the semantic interpretations and not the surface words. However, for the current stage of the project and given the inherent difficulty of automatic semantic annotation, we consider this to be a close approximation to the original polarity marking theory.

(4.8) a. some: $= NP^{+}/N^{+}$

b. $\circ$farmer $:= N^{\circ}$

c. $\circ$walks $:= S^{\circ}\backslash NP$

We can now obtain the polarity marking of the constituents by simply performing the parsing of the sentence and following the polarity conventions decribed above. The final stage of this process is to apply the syntactic environment of the root sentence, which must always be positive. Thus $\circ$ is instantiated as $+$ and $\bullet$ as $-$.

(4.9)
$$\dfrac{\dfrac{\dfrac{\dfrac{\textit{Some}}{\mathsf{NP}^{+}/\mathsf{N}^{+}} \quad \dfrac{\circ\textit{farmer}}{\mathsf{N}^{\circ}}}{\mathsf{NP}^{+}(\textit{Some}+\textit{farmer})}>}{\mathsf{S}^{\circ}(\textit{Some}+\textit{farmer}\circ\textit{walks})}< \quad \dfrac{\circ\textit{walks}}{\mathsf{S}^{\circ}\backslash\mathsf{NP}}}{\mathsf{S}^{+}(\textit{Some}+\textit{farmer}+\textit{walks})}$$

After this last step in the derivation, all the constituents are marked either as $+$ or $-$ which corresponds to an upward or downward direction of entailment. Using the result of the derivation in 4.9 we can safely draw the following upward entailments.

(4.10) a. Using [**farmer**]$\leq$[**person**] $\vdash$ Some person walks.

b. Using [**walks**]$\leq$[**moves**] $\vdash$ Some farmer moves.

c. Using [**Some farmer**]$\leq$[**Somebody**] $\vdash$ Somebody walks.

d. Using (b) and (c) $\vdash$ Somebody moves.

As we can see from example 4.8, common nouns bear a polarity-preserving mark, meaning that their determiner has to specify their final polarity. In the case of proper nouns, however, we cannot allow the polarity to be inverted. The reason is that proper nouns don't have any "smaller" semantic denotations. To address this issue we assign a $+$ mark to any noun category bearing a NNP (or NNPS) part-of-speech tag.

4.2.1 Polarity Inverting Terms

Under normal circumstances (and since the sentence environment is always positive) every entailment will be upward-monotone. However there are certain words (and expressions) that impose an inversion of polarity over the constituents that they dominate. The most prominent of these *polarity-inverting terms* is the negative particle **not** (and

it's contracted form **n't**). Although it is more common to use a single polarity-inverting category verbs like **doesn't**, since we are using a statistical parser that creates a separate token entry for **n't**, we must assign the polarity-inverting properties to the particle which would then be passed on to the auxiliary **do** (or **be**, **has**, **can** etc.).

This is achieved by using the new negation rule described earlier and the *neg* primitive type for the not and n't particles. The end result is the same with the category that Sánchez-Valencia (1991) uses for **doesn't**:

(4.11)
$$\dfrac{\dfrac{\textit{do}}{(S^{\circ}\backslash NP)/(S\backslash NP)} \quad \dfrac{\textit{n't}}{\mathsf{neg}}}{(S^{\circ}\backslash NP)/^{\bullet}(S\backslash NP)}{>}$$

Using this category we can analyse sentences with downward-monotone entailments in a similar way to that described earlier.

(4.12)
$$\dfrac{\dfrac{\dfrac{\textit{Some}}{NP^{+}/N^{+}} \quad \dfrac{\circ\textit{farmer}}{N^{\circ}}}{NP^{+}(\textit{Some}+\textit{farmer})}{>} \quad \dfrac{\dfrac{\textit{doesn't}}{(S^{\circ}\backslash NP)/^{\bullet}(S\backslash NP)} \quad \dfrac{\circ\textit{walk}}{S^{\circ}\backslash NP}}{S^{\circ}\backslash NP(\textit{doesn't}\bullet\textit{walk})}{>}}{S^{\circ}(\textit{Some}+\textit{farmer doesn't}\bullet\textit{walk})}{<}$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$S^{+}(\textit{Some}+\textit{farmer doesn't}-\textit{walk})$$

Negative polarity items (NPIs) are indicative of downward monotonicity, as Sánchez-Valencia (1991) and Dowty (1994) observe. In particular determiners like **any** and **no** licence a downward-monotone entailments.

(4.13) any : $= NP^{-}/N^{-}$

(4.14) no $:= NP^{\bullet}/N^{\bullet}$

NPIs may be considered as a closed word class, which means that they are easy to identify manually. In particular we have defined a list of known NPIs (only as surface words) which we use to assign the polarity-inverting (or $-$) marks to their arguments. However, there is another category of words with polarity-inverting properties that is not so easy to circumscribe. They are verbs that denote negation especially in communication. The most common example is **deny** which is defined as follows:

(4.15) $\circ$deny $:= (S^{\circ}\backslash NP)/S^{\bullet}$

Other polarity-inverting verbs include doubt, disbelieve, disprove, reject, refuse, contradict etc. Again, we have specified a list of these verbs created manually and on an intuitive basis. However, this is almost certainly an open class of words and identifying every possible term manually is an extremely difficult (if not impossible) task. One possible solution might be to implement a "word learner" that could identify the polarity mark of word by using contextual cues.

Further investigation is required to examine the coverage that our current list provides, as well as whether informative contextual cues exist[10].

Having identified the polarity of NPIs and polarity inverting terms, we assign every other constituent the most permissive polarity ($\circ$). This way all the expressions in a sentence will be upward-monotone except when a constituent specifically imposes a "stronger" polarity (whether it is $\bullet$ or $-$).

Finally, for our implementation we have made a simplification by assuming that polarity-inverting (and $-$) constituents apply the same polarity to all their arguments. For instance while the nominative type-raised version of **any** should be $(S^{-}/(S^{-}\backslash NP^{-}))/N^{-}$ we will treat it as $(S^{-}/^{-}((S\backslash NP))/N)$. Since (apart from their type-raised forms) most of the constituents in question don't have complex argument lists, this simplification seems reasonable[11].

Figure 4.4 presents a polarity-marked derivation of CCGbank (section 00) sentence adapted here for brevity[12]. We can see the application of the negation rule as well as the effect that it has on the subsequent constituents (for reasons of perspicuity we have annotated the surface words only at the final level of the derivation and we have left out most polarity-preserving derivations).

One might argue that this derivation is not entirely correct since the adverb **federally** is tagged as NP/NP while the corect category should be $(N/N)/(N/N)$. However as we have already stressed, the grammatical "correctness" of the parse is of little importance, as long as the relations between the constituents provide a correct environment for polarity marking.

[10]There are cases of polarity inverting terms which are even more difficult to process. One example is the phrase **I'm having second thoughts**. While the phrase as a whole imposes a polarity inversion, none of the individual constituents can bear a $\bullet$ mark. One way around this problem would be to assign a polarity mark once the complete derivation of the phrase emerges; however for now we will ignore this issue.

[11]In fact the only exception that we could find was reported in Dowty (1994), where the polarity marked category for **every** is $(S^{+}/(S^{+}\backslash NP))/N^{-}$ with positive and negative marks alternating in the arguments. However this category cannot be found in CCGbank corpus (where the most probable is NP/N) which makes it safe to assume that we will not have to confront this problem.

[12]The original sentence was: "He said the ban won't stop privately funded tissue-transplant research or federally funded fetal-tissue research that doesn't involve transplants."

Figure 4.4: Stat-OpenCCG derivation with polarity marking

4.3 Natural Logic Inference

We have now defined the monotonicity for all the constituents and we can start producing natural logic entailments. We have already seen (section 2.2) that we can produce entailments for a variety of constituents (like verbs, common nouns, noun phrases and even connectives). Unlike MacCartney and Manning (2007) we will focus on common nouns and verbs, as they are the most information-rich elements of the sentence. Since the ultimate goal for our system will be to provide input for large "common sense" knowledge bases like Cyc, we have to ensure that we include as much information as possible. A replacement of a determiner like **every** with **some** has a significant impact on theorem provers but it will provide little information about the (pragmatic) relations of the sentence concepts. On the other hand, in a sentence like "The band was marching", the replacement of **marching** with **walking**, while semantically insignificant, might help a knowledge base to identify **band** as a group of people (which have the ability to walk) instead of a range of frequencies.

4.3.1 WordNet Interface and Sense Disambiguation

In order to gain access to broader or narrower concepts we have to use a computational lexicon that contains semantically ordered entries. We have chosen to use WordNet for this task.

WordNet is currently the largest semantic lexical database for English (Fellbaum, 1998). It contains more than 150.000 words organized in over 115,000 *synsets*[13] for a total of 207,000 word-sense pairs (WordNet, 2008). Most synsets are connected to other synsets via a number of semantic relations. The two relations we are interested in are *hypernyms* and *hyponyms*[14]:

- Y is a hypernym of X if every X is a (kind of) Y (*canine* is a hypernym of *dog*)
- Y is a hyponym of X if every Y is a (kind of) X (*dog* is a hyponym of *canine*)

It is fairly intuitive that hypernyms correspond to upward entailing substitutions and hyponyms to downward substitutions.

[13] A synset (or synonym ring), is a group of data elements that are considered semantically equivalent for the purposes of information retrieval.

[14] Only common noun can relate via hyponyms; verbs can also relate via *troponyms* (specifying the manner in which a verb is being performed) but this in not a direct analogue to hyponyms. Further investigation is needed to determine whether we can use troponyms in a downward entailing environment.

However, in order to chose which hyper/hyponyms we can use for our substitutions we have to identify the correct sense of the word in question; we have to deal with semantic ambiguity. As we have already mentioned sense ambiguity can be either of a homonymic or a polysemic nature. Where homonymy is very easy to deal in the current project –since the parser provides us with the part-of-speech tags for each word– polysemy remains a difficult challenge. One of the main problems here, is that WordNet uses very fine-grained sense distinctions.

As we have seen in section 3.3, there are numerous techniques available for sense disambiguation, the most successful of which use WordNet as their main source. However since our final goal is to use the output of our system with a large knowledge base like Cyc to perform the logical inference, we propose that the best approach to resolving the ambiguity of the entailing terms is to integrate it with the inference process itself. This way the knowledge base (and the inference engine) will have total control over the sense it chooses, which should lead to a better accuracy.

More specifically we propose an inference engine that uses the following algorithm:

Use the knowledge base to reason about the original sentence.
if there there are unknown concepts:
  apply the polarity marking algorithm to obtain the monotonicity direction for the nouns and verbs of the sentence.
  for each entailing constituent:
    Obtain the list of possible senses from WordNet.
    for each of the senses:
      replace the constituent with the hyper/hypo-nyms related to the sense.
      Use the knowledge base to reason about the edited sentence.
      if there there are unknown concepts:
        continue

The intuition behind this idea is that only by using the correct sense of the word can the knowledge base produce a valid inference.

Chapter 5

Results

5.1 Introduction

We now turn to evaluate the performance of our system by examining each of its components separately. The reason for this is that while the system produces natural logic inferences –that is, sentences with more relaxed or specific concepts than the original sentence– the value of the sentences themselves cannot be estimated unless they are fitted into a wider inference system designed for a specific purpose with available evaluative methods and corpora (e.g. Textual Entailment). However, as we have already mentioned we have built this system as a proof-of-concept for the theory of natural logic and more specifically for the new implementation methodology that we propose.

We will therefore evaluate each of the components of our system so that we can have an indication of the overall quality of a complete application based on it.

5.2 Parsing

Since computational parsing applications have been in existence for more than 40 years, there is an abundance of parsing evaluation metrics and testing corpora. Even in the case of CCG, which (at least in its statistical format) is still at an early stage, there are various evaluation metrics proposed that use the CCGbank corpus as a base.

We have already examined some of these metrics including Precision, Recall and lexical dependencies (see section 3.1.2). However, none of these can be used with our parser to produce comparable results. The main reason is the significant differences that that distinguish our system from the other statistical CCG parsers.

As we discussed in chapter 4 we have based our parser on the OpenCCG library

which uses only CCG rules to determine whether a derivation is acceptable or not. The two statistical CCG parsers we have examined do not use the same mechanism. Especially in the case of Hockenmaier (2003) the system is in essence a PCFG parser using CCG categories. This leads to cases where a derivation may be given some probability mass without being "CCG-legal"[1].

Moreover, we have created two new rules for dealing with coordination and negation that suited the specific needs of this project. The application of these rules will not lead to the same derivation structures as the gold standard parses in the CCGbank. The same problem is caused by the fact that we are not treating punctuation, apart from full stops.

These differences not only prevent us from using the standard evaluation metrics of Precision and Recall[2] but also are the main reasons for achieving significantly lower parsing accuracy (and coverage).

In order to test the efficiency of our parser we rely on two metrics; the coverage and the percentage of correct lexical categories. The coverage is an indication of how "wide-coverage" our parser can really be, obtained by measuring the number of sentences for which it can get a full parse. The lexical categories are an indication of the correctness of a derivation, in that (especially in our parser) if the lexical categories are correct then the derivation must be valid. This is because OpenCCG uses only the CCG rules to create a derivation after the assignment of the lexical categories. Since the CCG rules are given –and therefore must be correct– they can only produce correct parses given the right categories.

Using these two measures we performed a number of experiments, focusing on the component that could be parameterised the most –the supertagger. We tested the parser on the development section of the CCGbank using a number of different values for the β value and the word frequency cutoff k. The results are shown in table 5.1.

Before we comment about the overall results we must highlight the difference in the β values between our implementation and the supertagger of Clark and Curran (2004a) presented in section 4.1.2. Their supertagger achieved an accuracy of over 98% using a β value of 0.01. In our case, the need for higher sentence coverage forced us to use an order of magnitude smaller β values.

This need for smaller β values, in conjunction with the significant loss of accuracy,

[1] We have already examined the cases of punctuation and negation (see sections 4.1.4 and 4.1.5).

[2] One might argue that we could use the dependency structures as an evaluation method, since they don't contain any tree derivation information and hence are more uniform. However for the time being our system is not able to produce any dependency structures.

β value	k value	Coverage(%)	LexCat(%)
	$k = 5$	18.32	89.44
$\beta = 0.075$	$k = 20$	18.95	89.09
	$k = 100$	19.80	88.54
	$k = 5$	19.88	88.15
$\beta = 0.05$	$k = 20$	20.51	87.81
	$k = 100$	21.51	87.34
	$k = 5$	31.43	81.49
$\beta = 0.01$	$k = 20$	33.57	80.55
	$k = 100$	36.41	79.22
	$k = 5$	63.77	70.04
$\beta = 0.001$	$k = 20$	69.63	68.02
	$k = 100$	**75.86**	**65.64**

Table 5.1: Parsing results on CCGbank section 00

	Coverage(%)	LexCat(%)
Hockenmaier (2003)	99.00	91.50
Clark et al. (2002)	-	90.30*
Clark and Curran (2004b)	93.00	92.50
Stat-OpenCCG parser	72.58	65.31

Table 5.2: Parsing results on CCGbank section 23: All results are produced using a POS tagger except for those of Clark et al. (2002), who use a pre-tagged version of the corpus. Given the accuracy of the best POS tagger we can assume a reduction in accuracy of about 3-5%.

	Train Time (*hrs*)	Train Mem. Usage (*GB*)	Parse Time (*hrs*)	Parse Mem. Usage (*GB*)
Clark and Curran (2004b)	17	31	0.16	-
Stat-OpenCCG parser	2.8	1.5	9.97	1.5

Table 5.3: Training and parsing performance statistics

reveal the significance of the parser's shortcomings. The results in table 5.1 show that the parser, when using mostly "normal" categories[3], can deal quite successfully with the "easy" sentences of the corpus (lexical category accuracy $\sim$ 89.5% but only for 18% of the sentences). However, mainly because of the lack of punctuation treatment, in order to obtain full derivations for the more difficult (or just lengthier) sentences, the parser must use more complex (and therefore less probable) categories which results in significantly lower accuracy.

The differences become obvious when we compare our best results with those reported in the other statistical CCG parsers (table 5.2). Both coverage and accuracy are significantly lower than the current state-of-the-art. These results, however, are slightly deceptive. In our parser the lexical categories selected for a sentence can only lead to CCG-legal derivations; the assignment of correct lexical categories may lead, in lack of an appropriate CCG rule, to a failed parse. If we take this fact into consideration, we can assume a smaller relative difference in terms of accuracy.

The last point of comparison we will examine is the computational performance where the only point of reference available is the parser of Clark and Curran (2004b). The results are presented in table 5.3. Both parsers use the same CCGbank sections for training (sections 01-22) and testing (section 23).

These results show clearly the run-time-optimised approach of Clark and Curran (2004b). They perform a time and resource-intensive training[4] of their discriminative model, which results in extremely fast parse times ($\sim$ 4 sentences/second). On the other hand, our parser performs much slower in run-time –although a significant factor for this performance is the implementation language– while training requires less than half the time and less than 1/20th of the memory. The difference between the generative and discriminative approaches in training time is even more striking if we consider that, out of the 2.8 hours needed for training, more than 2.7 hours are required for the training of the supertagger (which uses a log-linear/discriminative model) and only 2 minutes for the training of the generative model of the parser.

[3]Normal here is used to signify the few most probable categories that the narrow threshold of $\beta = 0.075$ will allow. These are the categories that a human annotator would probably suggest, hence the term "normal".

[4]The differences in training time are even more striking, considering that Clark and Curran (2004b) are using a 64-node Beowulf cluster while we trained on a dual-core Inter Celeron at 1.8GHz with 2GB RAM.

5.3 Natural Logic Inferences

Evaluating the Natural Logic part of our system presents a significant challenge, as there are no objective means of evaluation, let alone testing corpora. This is partly because most studies in Natural Logic examine only fractions of the English grammar, but also because identifying the correct polarity of a constituent, especially in negative concord environments, can be quite subjective.

For these reasons we cannot define a large-scale automated evaluation method; instead we will present our initial findings based on a manual evaluation of a small set of sentences[5]. We have randomly selected 100 sentences from the 00 section of the CCGbank (sentence IDs 03.30–18.15) –although we had to use 48 more sentences to get to 100, because of the parsing coverage. The results are presented in the first row of table 5.4. The table shows only the polarity marks of the verbs and common nouns.

	Sentences	**Sentences**	**Polarity Marks**		**Accuracy**	
	$(total)$	$(parsed)$	$(+)$	$(-)$	$(+)$	$(-)$
NO PUNCT.	148	100 (62%)	100	14	100%	(10/14) 71.4%
COMMAS	209	100 (48%)	100	11	100%	(11/11) 100%

Table 5.4: Polarity marking results

The first observation is that negative polarity expressions are quite uncommon, occurring only 9% of the time. This appears to have an intuitive basis, as most everyday sentences are declarative (or affirmative).

All 4 errors refer to $(-)$ mark insertions and all of them are due to derivation errors. More specifically these errors are caused by the incorrect treatment of commas. The sentences in these examples contain independent clauses that, without correct treatment of commas, are attached to the main body of the sentence and therefore within the scope of the polarity inverting term.

For instance, in sentence 5.1[6], we can see that the verb **said** and the nouns **officer** and **interview** have been marked erroneously as $(-)$ because the comma after "choice" that should separate the two clauses, has been ignored.[7].

[5] We have already examined a successful polarity marking of the sentence 47.5 of the 00 section of CCGbank in figure 4.4.

[6] The original sentence was: "We did n't have much of a choice, Cray Computer 's chief financial officer, Gregory Barnum, said in an interview."

[7] This sentence contains 3 of the total 4 negative polarity errors. The fourth error occurred in the

(5.1) We (+)did n't (−)have much of a (−)choice Cray Computer 's chief financial (−)officer Gregory Barnum (−)said in an (−)interview .

To illustrate the importance of punctuation treatment –especially commas– we have included another set of results in which the parser uses the very simple punctuation rule presented in section 4.1.4 for both commas and full stops. These results are shown in the second row of table 5.4.

As we can see, while the coverage of the parser decreases dramatically, the accuracy of the (−) polarity marks rises to 100%. This happens because, in spite of being far more difficult to obtain a full derivation from a sentence containing commas, when such a sentence is parsed, we can be sure that the independent clauses remain outside the scope of the polarity-inverting term.

5.4 Using Natural Logic Inferences in Cyc

We have already mentioned that one of the goals of this project is to provide input for higher-level NLP systems for applications such as Textual Entailment Recognition. Instead of directly applying the natural logic inferences (as MacCartney and Manning, 2007 do) we will focus on using these inferences to increase the lexical coverage of "deep" reasoning tools. One of the most promising systems in this field is *Cyc* (Lenat, 1990), a Very Large-Scale Knowledge Base (VLKB) that contains "common-sense" concepts and assertions that the average person knows about the world. Cyc also incorporates an inference engine that allows it to reason about the relations between concepts.

These features attracted many researchers to use Cyc as an inference engine for Natural Language applications; however most of them discovered important drawbacks that hindered their efforts. The most common problem was the lack of specific concepts from Cyc's ontology that were crucial for the inference process. The reports of Mahesh et al. (1996) and Cox (2005) verify this problem, even though they were created more than 9 years apart (signifying the persistence of the problem).

Our initial plan was to evaluate our system by providing evidence of improved concept coverage in the some of the cases reported by Cox (2005) –since this was

sentence "The (+)governor could n't (−)make it so the (+)lieutenant (+)governor (−)welcomed the special (+)guests . ", which contains the same type of error (the comma would normally separate the clause after **it**).

the latest large-scale study, based on the RTE corpus[8]. However, it seems that in the version of Cyc used for our experiments, most of the coverage problems reported by Cox (2005) have been resolved. The following examples present these improvements.

(5.2) RTE index: QA 594

T: *"For the first time in history, the players are investing their own money to ensure the future of the game", Atlanta Brave pitcher Tom Glavine said.*

H: Tom Glavine plays for the Atlanta Braves.

Cox (2005) reports that the concept *Pitcher-TheWord* had only one denotation, and it is a *SevingVessel*, not a *BaseballPlayer*.

New Additions:

BaseballPitcher, genls[9] : *BaseballPlayer*

The relation that connects the concept of BaseballPlayer with the activity of Playing is also included:

(*prototypicalActivityTypeOfPersonType*
(PlayingFn *Baseball-TheGame*) *BaseballPlayer*)

(5.3) RTE index: QA 1554

T: *In fact, Woolsey had had no first-hand experience with the world of spies until President Bill Clinton appointed him Director of Central Intelligence.*

H: James Woolsey is the director of the CIA.

Cox (2005) reports that the only relevant translation of *appoint-TheWord* is related only with the concept of *AppointingAmbassador* and therefore Cyc can't handle the event of "director appointment".

New Additions:

Appointment, genls : *PositionSelection*

which contains the following assertion, signifying the relationship between the position and the action of appointment:

(implies
(and
(*isa* ?SEL *PositionSelection*)
(*chosenItem* ?SEL ?THING)
(*positionToFill* ?SEL ?POS ?ORG))

[8]The RTE corpus is a test set of textual entailment pairs (premise-hypothesis), provided for the purposes of the Recognising Textual Entailment Challenge (Dagan et al., 2006).

[9]*genls* signifies the generalisation relation between two concepts.

(*chosenFor* ?SEL ?THING
(*positionOfPersonInOrganization* ?THING ?ORG ?POS)))

(5.4) RTE index: CD 767
T: *Hepburn, a four-time Academy Award winner, died last June in Connecticut at age 96.*
H: Hepburn, who won four Oscars, died last June aged 96.

Cox (2005) reports that there is no entry for "Academy Award" or "Oscar" in Cyc.
New Additions:
Oscar-TheWord that contains the following denotation:
(denotation *Oscar-TheWord ProperCountNoun* 0 *AcademyAward*)

While some of these additions may have been cases of routine knowledge base expansions, one might suspect that most of the new concepts were created in response to the reports of Cox (2005) and others. For whatever reason, these additions imply that we cannot draw useful conclusions based solely on the RTE data.

This doesn't mean that Cyc, in its present version, contains all the concepts necessary for textual entailment. To show that Cyc still suffers from lexical coverage problems –and at the same time how can our system provide essential information to the inference process– we will use the following example we constructed.

(5.5) **T**: *John had to perform a thoracotomy yesterday.*
H: John is a surgeon.

Cyc contains no entry for the concept *thoracotomy*. Using our system we obtain the following analysis for the premise text:

(5.6) John $(+)$had to $(+)$perform a $(+)$thoracotomy $(+)$yesterday.

Since Cyc contains every other concept of the sentence, we will use the hypernym relation of WordNet to relax the sense of **thoracotomy** to **incision**[10]. Cyc recognises the concept *SurgicalIncision* from the following denotation:

(5.7) (denotation *Incision-TheWord CountNoun* 0 *SurgicalIncision*).

By examining the concept of *SurgicalIncision*, Cyc knows that it occurs at a *MinimallyInvasiveSurgery* from the assertion:

[10]WordNet contains only one sense for thoracotomy therefore there is no need to apply our WSD algorithm.

(5.8) (and
(*isa* ?INCISION *SurgicalIncision*)
(*eventOccursAt* ?SURG ?INCISION)
(*isa* ?SURG *MinimallyInvasiveSurgery*))

From the concept of *MinimallyInvasiveSurgery*, Cyc can get to its generalisation, which is the concept of *Surgery*. Finally within the concept of *Surgery*, Cyc can find the relation to the concept of *Surgeon*:

(5.9) *frequentPrototypicalActivityTypeOfPersonType* : *Surgeon*

The inference is now complete, since it is trivial for Cyc to recognise **John** (or any other Proper Name) as a *Person* concept.

Although this example was quite simple, we can see that logical inference using world knowledge is not trivial and it is important to increase the amount of information available to the inference engine. To this end, Natural Logic inferences of the type our system produces, provide a valuable service, though the full extent of the contribution depends on the specific NLP task.

Chapter 6

Discussion

6.1 General Remarks

In this project we have examined the creation of a natural logic inference system with respect to its various components. We shall now provide a critical evaluation of this work and in the next section we will examine further expansions and adaptations of the system.

The main contributions of this work were the creation of a statistical version of the OpenCCG parser and the implementation of an alternative monotonicity marking scheme integrated with the parsing process.

In the case of the statistical-OpenCCG, although our work was concentrated in the creation of a suitable environment for the polarity marking, we have shown that the conversion is possible even by using a very simple statistical model. We have to emphasise OpenCCG's approach to the parsing process where, by only allowing CCG rule-permissible derivations, it creates a grammatically consistent environment. While this might lead to inferior performance (see section 5.2) we can speculate that "cleaner" training corpora and higher quality lexicons will significantly improve the parsing accuracy.

We believe that lexical acquisition is a vital part of high-quality parsing, especially in lexical grammars like CCG. The reason is that all the syntactic ambiguity in CCG comes from the lexical categories. If one could define unambiguously (or with low ambiguity) the categories for every constituent in a sentence, there would be no reason for having a statistical model (or at least an elaborate one). Therefore we believe that shifting our attention from high-quality parsing to high-quality lexical acquisition is the key for the next generation of wide-coverage parsing.

Nevertheless, this is not an attempt to diminish the importance of the current parsing techniques and statistical models. CCGbank is the only large-scale CCG corpus available at present, and we should focus our attention towards getting the best results with it. In this frontier our parser achieves only moderate results compared to the other statistical CCG parsers and our immediate priority is to ameliorate these results. In the next section we will discuss a number of possible improvements to this end.

The new approach to monotonicity marking that we implemented, seems to provide accurate results even with the modifications of the theory proposed in Steedman (2008). As we have already discussed, there is an inherent difficulty in obtaining quantitative results since there is no testing corpus available and most of the work in the field has used only a fraction of the English grammar (see section 3.2).

The final step of the natural logic inference is to either expand or narrow the meaning of the selected constituents (in this case common nouns and verbs) according to the directionality of their monotonicity. We have chosen to use the hyper/hyponymic relations of the WordNet database as it is the most comprehensive (structured) lexical database to date. This of course meant that we had to choose the correct sense of the given word within WordNet. In section 4.3.1 we examined an approach that integrates the WSD process into that of entailment recognition, by using the knowledge base as a source for both. Despite the fact that this is only a theoretical construct, the work of Curtis et al. (2006) and the preliminary results about the concept coverage of Cyc (discussed in section 5.4) provide significant evidence to the success of this method.

One way for evaluating this implementation as a whole would be to analyse the results from a higher-level task (such as Textual Entailment) where the monotonicity calculus could provide correct (or erroneous) word-level inferences. This method of inference may not be adequate by itself for a textual entailment task, because as MacCartney and Manning (2007) point out, the related corpora do not provide an environment suitable for Natural Logic inferences. This means that the sentences in a textual entailment corpus contain very few instances of polarity inverting terms, or $(+)$ assigning terms (like existential quantifiers). Nevertheless, since our approach focuses on common noun and verb replacements (instead of MacCartney and Manning's NP replacements), the results on this task might have been different. However, in the current stage of development, our system cannot be used in textual entailment recognition for it lacks the required tools; tools like syntactic matching/alignment or theorem provers/model builders are instrumental for shallow or deep reasoning respectively.

Therefore we have decided to examine the efficiency of our system in resolving

Cyc's lexical coverage problems as reported by Mahesh et al. (1996) and Cox (2005). However, as we saw in section 5.4, this has proven to be more difficult than expected, mainly because of improvement of the lexical coverage in the latest version of Cyc used for our evaluation. This, of course, does not mean that Cyc contains every possible concept; as we saw in example 5.5, Cyc has got no entry for *thoracotomy*, while it contains the names of other surgical incisions (e.g. prostatectomy, tonsillectomy and vasectomy can be found as instances of the *Surgery* collection). In this case our system was able to replace the specific term with its direct WordNet hypernym (*incision*), which was contained in Cyc's ontology and had the semantic relations that could lead to the successful recognition of the entailment.

6.2 Future Work

We view the work described here as an initial effort towards a full-fledged inference system designed specifically for textual entailment. With this perspective in mind we propose a number of possible improvements and additions to the current system.

Currently the most important drawback in our system is the low parsing performance on the CCGbank. To improve this we must address several issues, most important of which is the treatment of punctuation. The most common and most difficult punctuation mark to treat is the comma (,). The comma has several uses in the English grammar, making it highly ambiguous. It can be used among others (Wikipedia, 2008):

- To mark introductory words and phrases: "*Once upon a time,* I didn't know how to use commas."
- To mark parenthetical phrases: "John, *after having a drink*, walked home."
- To separate items in lists: "I like *apples, bananas, grapes* and oranges."
- To indicate that a word has been omitted: "The cat *was* white; the *dog, brown*."
- To set off quoted material: "Mr Smith says, *You must go home*."

All these different uses[1] are given the same category in the CCGbank and it is only

[1]There is another (optional) use of the comma in which it can remove the syntactic ambiguity (e.g. in prepositional phrase attachment). If we consider the following examples we can see that in 6.2 the ambiguity is resolved and John is using the telescope to see the man:

(6.1) John sees the man with the telescope.

(6.2) John sees the man, with the telescope.

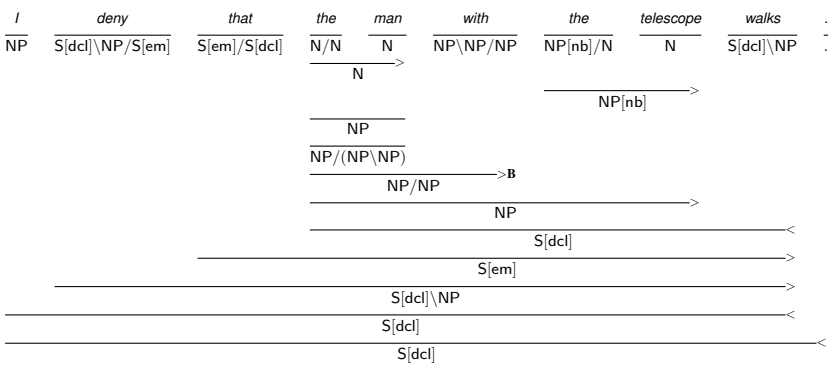
because of existing training data that the other two CCG parsers can produce correct derivations that contain commas. With the current setup of our parser –unifying constituents only if it is allowed by some CCG rule– it is impossible to create correct derivations unless we can identify the different uses of the comma and assign to each one the appropriate CCG category.

Similar problems arise when dealing with other punctuation marks, although none of them has the ambiguity of the comma. The optimal solution for this situation is to assign proper CCG categories to every punctuation mark, reflecting its syntactic use in the sentence.

To do this we have to rely on contextual cues that uniquely identify which of the possible uses appears in a given sentence. For instance we can easily identify the list separation usage if we notice that the constituents on either side of the comma have the same categories. Although finding such cues for all the comma uses might not be easy, we know that similar methods are used in lexical acquisition (Thomforde, 2008) and they present a reasonable starting point.

Treating punctuation is only one side of the problem, though. It has been shown by Hockenmaier (2003) and Clark et al. (2002) that capturing word-word dependencies significantly increases the performance of the parser. Moreover, these dependency structures could be used to provide better information about the effect of polarity inverting terms. This is especially true for long-distance (or deep) dependencies where the affected constituent might be a significant distance from the polarised term. The problem in this case is not that we can't reach the constituent in question but that in the process we will impose a polarity change on constituents that should remain unaffected. For instance consider the following example:

(6.3)



In this case **deny** should invert the polarity of **walks**; however, in the current implementation **telescope** will be affected too, leading to an erroneous inference. Creating a dependency-based parser will provide a solution to this problem.

Further modifications to the parser should include the addition of interpolation between the conditional probabilities and back-off techniques for unknown derivations.

From a computational point of view, we also have to address the issue of efficiency, since the parser performs considerably slower than the times reported in Clark and Curran (2004b) and the resource demands are quite substantial. The main issue here is memory allocation, which could be provided through an efficient pruning technique.

Moving on to the natural logic inferences, there are two main problems. The first is that we have only identified a limited number of polarity inverting terms. We believe that they constitute an open class of words (except perhaps from the standard NPIs) and thus are very difficult to circumscribe. We have mentioned that one possible solution for this is to use a "word learner" that would predict the polarity of a word given its contextual cues. However most (or even all) word learners use syntactic information as cues (Thomforde, 2008) which are not indicative of polarity behaviour. One could apply a word learner on a semantically (or even better, polarity) annotated corpus to get more relevant cues, but such corpora (to our knowledge) do not exist in large scale.

The second problem concerns WordNet's sense ambiguity. We have presented an ambiguity resolution algorithm using the inference system itself, and the preliminary results show its effectiveness. Nevertheless we still believe that sense distinction in WordNet is sometimes artificially fine-grained. In Senseval-2, the second WSD challenge, researchers proposed the grouping of WordNet senses to provide a more coarse-grained sense distinction (cf. Palmer et al., 2007, pp. 95–98). We should examine if this is a suitable approach for this project as well.

Finally we should consider the adaptation of the parser to include modality features and semantic interpretations. While neither of these additions will lead to significant improvement in terms of parsing accuracy, they might prove to be important for the polarity marking and logical inferences. Modalities will give the lexicon more control over the grammar rules, which will probably lead to better derivation structures. Small-scale tests that we performed with hand-built "toy" grammars, provide a solid indication for this direction. However, automatically inducing the modality of each category slash is not trivial and it will almost certainly require a manually annotated modality corpus for lexical training.

Semantic information in the form of λ-calculus or OpenCCG's *hybrid logic* will provide an intermediate step between the surface structure and Cyc's higher-order logic calculus. Using first order logic expressions in an external model builder where Cyc only provides the background knowledge might be more effective than using Cyc's

internal (and more brittle) inference engine to reason about the data. Again, the main difficulty is to automatically assign semantic interpretations to the lexical entries. Bos (2005) presents an approach for semi-automatically constructing semantic interpretations of CCG derivation structures (using the parser of Clark and Curran (2004b) as a base), but his research is still at an initial stage and the results are not optimal.

Although there is a considerable amount of work to be done, in this project we have shown that it is possible to create a wide-coverage Natural Logic inference system using the tools provided by Combinatory Categorial Grammar, and that these inferences can be used as input to a larger Natural Language system with encouraging results.

Bibliography

Agirre, E. and Edmonds, P. (2007a). Introduction. In Agirre, E. and Edmonds, P., editors, *Word Sense Disambiguation: Algorithms and Applications*, pages 1–28. Springer.

Agirre, E. and Edmonds, P., editors (2007b). *Word Sense Disambiguation: Algorithms and Applications*, volume 33 of *Text, Speech and Language Technology*. Springer.

Ajdukiewicz, K. (1935). Die syntaktische konnexität. In McCall, S., editor, *Polish Logic 1920-1939*, pages 207–231. Oxford University Press, Oxford.

Baldridge, J. (2002). *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.d., University of Edinburgh.

Baldridge, J. (2008). List of projects which use OpenCCG. http://comp.ling.utexas.edu/wiki/doku.php/openccg/projects_using_openccg. [Online; accessed 10-August-2008].

Baldridge, J. and Kruijff, G.-J. (2003). Multi-modal combinatory categorial grammar. In *Proceedings of the 11th Annual Meeting of the European Association for Computational Linguistics*, pages 319–326, Philadelphia, PA.

Bangalore, S. and Joshi, A. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Bar-Hillel, Y., Gaifman, C., and Shamir, E. (1964). On categorial and phrase structure grammars. In Bar-Hillel, Y., editor, *Language and Information*, pages 99–115. Addison-Wesley, Reading, MA.

Black, E., Abney, S., Flickinger, D., Grishman, C. G. R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 306–311, San Mateo, CA.

Bos, J. (2005). Towards wide-coverage semantic interpretation. In *Proceedings of the Sixth International Workshop on Computational Semantics (IWCS-6)*, pages 42–53, Tilburg, Netherlands.

Bos, J. and Markert, K. (2006). When logical inference helps determining textual entailment (and when it doesn't). In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 98–103, Venice, Italy.

Charniak, E. (1996). Tree-bank grammars. In *Proceedings of the Thriteenth National Conference on Artificial Intelligence (AAAI '96)*, pages 1031–1036, Portland, OR.

Chomsky, N. (1957). *Syntactic Structures*. Mouton, The Hague.

Clark, S. (2002). Supertagging for combinatory categorial grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6)*, pages 19–24, Venice, Italy.

Clark, S. and Curran, J. (2004a). The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 282–288, Geneva, Switzerland.

Clark, S. and Curran, J. R. (2004b). Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 103–111, Barcelona, Spain.

Clark, S., Hockenmaier, J., and Steedman, M. (2002). Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 327–334, Philadelphia, PA.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, Santa Cruz, CA.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. Ph.d. thesis, Computer and Information Science, University of Pennsylvania.

Collins, M. J. (1997). Three generative lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain.

Cox, C. (2005). Assessing the utility of researchcyc in recognizing textual entailment. Technical report, Department of Computer Science, Stanford University.

Curry, H. B. and Feys, R. (1958). *Combinatory Logic: Vol. I*. North Holland, Amsterdam.

Curtis, J., Cabral, J., and Baxter, D. (2006). On the application of the cyc ontology to word sense disambiguation. In *Proceedings of the Nineteenth International FLAIRS Conference*, pages 652–657, Melbourne Beach, FL.

Dagan, I., Glickman, O., and Magnini, B. (2006). The pascal recognising textual entailment challenge. In J. Quiñonero-Candela and I. Dagan and B. Magnini and F. d'Alché-Buc, editor, *Machine Learning Challenges*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer-Verlag, Heidelberg, Germany.

Dowty, D. (1994). The role of negative polarity and concord marking in natural language reasoning. In *Proceedings of the Fourth Conference on Semantics and Linguistics Theory (SALT IV)*, pages 114–144, Rochester, NY.

Edmonds, P. (2006). Lexical disambiguation. In Brown, K., editor, *The Elsevier Encyclopedia of Language & Linguistics*, pages 607–623. Elsevier, 2nd. edition.

Eisner, J. (1996). Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34nd Annual Meeting on Association for Computational Linguistics*, pages 79–86, Santa Cruz, CA.

Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA.

Fyodorov, Y., Winter, Y., and Francez, N. (2003). Order-based inference in natural logic. *Logic Journal of the IGPL*, 11(4):385–416.

Gale, W. A., Church, K. W., and Yarowsky, D. (1992). A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26(5–6):415–439.

Gazdar, G. (1988). Applicability of indexed grammars to natural languages. In Reyle, U. and Rohrer, C., editors, *Natural Language Parsing and Linguistic Theories*, pages 69–94. Reidel, Dordrecht.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. (2007). The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague, Czech Republic.

Guthrie, J. A., Guthrie, L., Wilks, Y., and Aidinejad, H. (1991). Subject-dependent co-occurrence and word sense disambiguation. In *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*, pages 146–152, Berkeley, CA.

Halliday, M. A. K. (1994). *An Introduction to Functional Grammar*. Hodder Arnold.

Henderson, J. (2003). Generative vs. discriminative models for statistical left-corner parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 115–126, Nancy, France.

Hirst, G. (1987). *Semantic Interpretation and the Resolution of Ambiguity*. Studies in Natural Langugage Processing. Cambridge University Press.

Hockenmaier, J. (2003). *Data Models for Statistical Parsing with CCG*. Ph.d. thesis, School of Informatics, University of Edinburgh.

Hockenmaier, J. and Steedman, M. (2002). Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342, Philadelphia, PA.

Ide, N. and Véronis, J. (1998). Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics*, 24(1):1–40.

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

Joshi, A. (1988). Tree adjoining grammars. In Dowty, D., Karttunen, L., and Zwicky, A., editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge.

Krovetz, R. and Croft, B. W. (1989). Word sense disambiguation using machine-readable dictionaries. In *Proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 127–136, Cambridge, MA.

Lakoff, G. (1970). Linguistics and natural logic. *Sythese*, 22:151–271.

Lenat, D. B. (1990). *Building Large Knowledge-Based Systems : Representation and Inference in the Cyc Project*. Addison-Wesley, Reading, MA.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, pages 24–26, Toronto, Canada.

MacCartney, B. and Manning, C. D. (2007). Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200, Prague.

Mahesh, K., Nirenburgh, S., Cowie, K., and Farwell, D. (1996). An Assessment of Cyc for Natural Language Processing. Technical Report MCCS-96-302, Computing Research Laboratory, New Mexico State University.

Manning, C. D. and Schütze, H. (2000). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, 2nd edition.

Masterman, M. M. (1957). The thesaurus in syntax and semantics. *Mechanical Translation*, 4(1-2):35–43.

Mihalcea, R. (2007). Knowledge-Based Methods for WSD. In Agirre, E. and Edmonds, P., editors, *Word Sense Disambiguation: Algorithms and Applications*, pages 107–131. Springer.

Palmer, M., Ng, H. T., and Dang, H. T. (2007). Evaluation of wsd systems. In Agirre, E. and Edmonds, P., editors, *Word Sense Disambiguation: Algorithms and Applications*, pages 71–106. Springer.

Pollard, C. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, IL.

Ratnaparkhi, A. (1996). A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Philadelphia, PA.

Resnik, P. (1995). Disambiguating noun groupings with respect to wordnet senses. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 54–68, Cambridge, MA.

Roeck, A. D. (1983). An underview of parsing. In King, M., editor, *Parsing Natural Language*, pages 3–17. Academic Press, London.

Ross, J. R. (1967). *Constraints on Variables in Syntax*. Ph.d. thesis, MIT. Published as *Infinite Syntax!*, Ablex, Norton, NJ, 1986.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, Upper Saddle River, NJ, 2nd edition.

Sánchez-Valencia, V. (1991). *Studies on Natural Logic and Categorial Grammar*. PhD thesis, University of Amsterdam.

Sánchez-Valencia, V. (1995). Parsing-driven inference: Natural logic. *Linguistic Analysis*, 25:258–285.

Small, S., Cottrell, G., and Tanenhaus, M. (1988). *Lexical Ambiguity Resolution*, chapter Preface, pages 3–9. Morgan Kaufmann.

Steedman, M. (1996). *Surface Structure and Interpretation*. The MIT Press, Cambridge, MA.

Steedman, M. (2000). *The Syntactic Process*. The MIT Press, Cambridge, MA.

Steedman, M. (2008). The Grammar of Scope. Draft version.

Steedman, M. and Baldridge, J. (2007). Combinatory categorial grammar. In Borsley, R. and Borjars, K., editors, *Non-Transformational Syntax*. Blackwell. To appear, Draft 5.0, April 19, 2007.

Steedman, M. and Prevost, S. (1994). Specifying intonation from context for speech synthesis. *Speech Communication*, 15:139–153.

Sussna, M. (1993). Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the 2nd International Conference on Information and Knowledge Management*, pages 67–74, Washington D.C.

Thomforde, E. (2008). Semi-supervised Lexical Acquisition for Wide-Coverage Parsing. Ph D. Proposal, University of Edinburgh.

Toutanova, K., Klein, D., Manning, C., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Human Language Technology Conference (HLT-NAACL 2003)*, pages 252–259, Edmonton, Canada.

Toutanova, K. and Manning, C. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 63–70, Hong Kong.

van Benthem, J. (1991). *Language in Action*, volume 130 of *Studies in Logic and The Foundations of Mathematics*. North-Holland, Amsterdam, Holland.

van Eijck, J. (2005). Natural logic for natural language. www.cwi.nl/ jve/papers/05/nlnl/NLNL.pdf.

Véronis, J. and Ide, N. (1990). Word sense disambiguation with very large neural networks extracted from machine readable dictionaries. In *Proceedings of the 13th conference on Computational linguistics*, pages 389–394, Helsinki, Finland.

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

Vijay-Shanker, K. and Weir, D. J. (1994). The equivalence of four extensions of context-free grammars. *Theory of Computing Systems*, 27(6):511–546.

Weaver, W. (1955). Translation. In Locke, W. N. and Booth, A. D., editors, *Machine Translation of Languages*, pages 15–23. New York: John Wiley & Sons. Reprint of the 1949 mimeographed memorandum.

White, M. (2008). OpenCCG: The OpenNLP CCG Library. http://openccg.sourceforge.net. [Online; accessed 10-August-2008].

Wikipedia (2008). Comma (punctuation) — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Comma_(punctuation). [Online; accessed 10-August-2008].

Winograd, T. (1972). *Understanding Natural Language*. Edinburgh University Press.

Woods, W. A. (1973). An experimental parsing system for transition network grammars. In Rustin, R., editor, *Natural Language Processing*, pages 111–154. Algorithmics Press, New York.

WordNet (2008). Wordnet 3.0 database statistics. http://wordnet.princeton.edu/man/wnstats.7WN. [Online; accessed 12-August-2008].

Yarowsky, D. (1992). Word-sense disambiguation using statistical models of roget's categories trained on large corpora. In *Proceedings of the 14th Conference on Computational linguistics*, volume 2, pages 454–460, Nantes, France.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.