THE UNIVERSITY *of* EDINBURGH

Semi-supervised Lexical Acquisition for Wide-coverage Parsing

Emily Thomforde

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2012

Abstract

State-of-the-art parsers suffer from incomplete lexicons, as evidenced by the fact that they all contain built-in methods for dealing with out-of-lexicon items at parse time. Since new labelled data is expensive to produce and no amount of it will conquer the long tail, we attempt to address this problem by leveraging the enormous amount of raw text available for free, and expanding the lexicon offline, with a semi-supervised word learner. We accomplish this with a method similar to self-training, where a fully trained parser is used to generate new parses with which the next generation of parser is trained.

This thesis introduces Chart Inference (CI), a two-phase word-learning method with Combinatory Categorial Grammar (CCG), operating on the level of the partial parse as produced by a trained parser. CI uses the parsing model and lexicon to identify the CCG category type for one unknown word in a context of known words by inferring the type of the sentence using a model of end punctuation, then traversing the chart from the top down, filling in each empty cell as a function of its mother and its sister.

We first specify the CI algorithm, and then compare it to two baseline word-learning systems over a battery of learning tasks. CI is shown to outperform the baselines in every task, and to function in a number of applications, including grammar acquisition and domain adaptation. This method performs consistently better than self-training, and improves upon the standard POS-backoff strategy employed by the baseline StatCCG parser by adding new entries to the lexicon.

The first learning task establishes lexical convergence over a toy corpus, showing that CI's ability to accurately model a target lexicon is more robust to initial conditions than either of the baseline methods. We then introduce a novel natural language corpus based on children's educational materials, which is fully annotated with CCG derivations. We use this corpus as a testbed to establish that CI is capable in principle of recovering the whole range of category types necessary for a wide-coverage lexicon.

The complexity of the learning task is then increased, using the CCGbank corpus, a version of the Penn Treebank, and showing that CI improves as its initial seed corpus is increased. The next experiment uses CCGbank as the seed and attempts to recover missing question-type categories in the TREC question answering corpus. The final task extends the coverage of the CCGbank-trained parser by running CI over the raw text of the Gigaword corpus. Where appropriate, a fine-grained error analysis is also undertaken to supplement the quantitative evaluation of the parser performance with deeper reasoning as to the linguistic points of the lexicon and parsing model.

Acknowledgements

No PhD is achieved single-handed, and this one makes no pretence of exception.

I am grateful to Mark Steedman for his supervision, to Steve Clark and Laura Rimell for provision of the annotated QA corpus, to Christos Christodoulopoulos for the StatOpenCCG parser, to Tejaswini Deoskar for editorial advice, and to Luke Zettlemoyer for considerable mathematical assistance. Thanks are also due to my examiners, Sharon Goldwater and John Carroll, whose critical response has resulted in the improvement of this document.

A great deal of moral support was provided over many years by Jeri Crystal, Lynne and David Montgomerie, Joy Mills, Tommy Herbert, Annabel Harrison, Thom Scott-Phillips, and Amy Radermacher. I am also indebted to the many copyeditors of the peanut gallery: Elizabeth McDonald, Dave Mister, and the rest of the EFN. And mostly to my very helpful and very forgiving husband, Jamie Montgomerie, who not only put up with me throughout the process, but read the whole thing when I finished.

This work was partially funded by EU ERC Advanced Fellowship 249520 GRAM-PLUS and IST Cognitive Systems IP EC-FP7-270273 "XPERIENCE" and a grant from Wolfson Microelectronics.

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.

(*Emily Thomforde*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

> Imagine a language that is completely unfamiliar; the only means of studying it are an ordinary grammar book and a very large corpus of text. No dictionary is available. How can easily recognized, surface grammatical facts be used to extract from a corpus as much syntactic information as possible about individual words?
> Michael Brent [1993]

Natural language is varied and changing. It is full of errors, conventions, and fragments. Brent's formulation of the word-learning problem above is both abstracted and idealised. He envisions a problem space where we have a small amount of very specialised and error-free training data in the form of a lexicon and a large set of unlabelled sentences. Where once linguistics was comparatively rich in grammar books, its current wealth of information is in treebanks, and it is currently easier to acquire an inconsistently labelled corpus of sentences than to produce an exhaustive error-free grammar. We still view raw text as a free resource, but the scale on which is it available has exploded.

In the age of Google, Brent's challenge can be meaningfully reinterpreted as follows: Given a parser trained on a small amount of labelled data, and a very, very large corpus of text, how can the power of the parser be leveraged to extract as much information as possible about the lexicon? In seeking to address this task of lexicon extension, this thesis constructs and tests a word-learning system that leverages the known lexicon and statistical model to assign categories to out-of-lexicon (OOL) words in context.

1.1 Motivation from Linguistics

From a purely lexical perspective, many words can be categorised by contextual mutual intersubstitutability, meaning that if two words occur in identical contexts, they must share a syntactic category. This is certainly true for a large portion of the lexicon, since certain contexts uniquely license certain categories:

(1) I like to *run*.

(2) I like to *cook*.

(3) *I like to *potato*.

Here, the context *I like to X.* licenses the intransitive verbal category (represented as $S[b]\backslash NP$ in Combinatory Categorial Grammar (CCG)). No word that is not an intransitive verb ever occurs in this context, and conversely, any word in this context can be shown to have an intransitive sense among its uses. *potato* is unlikely to occur as above, but if it ever should, the lexicographer would be forced to classify it, if not a typo, as a new use of the word. These contextual patterns can indicate the category for a neologism, or previously unseen word, as in (4), or equally a new sense of a known word that already has at least one category, as in (5)[1].

(4) I like to *mosh*.

(5) I like to *text*.

However, we quickly begin to see see rarer constructions that defy this theory of contextual patterns:

(6) We have *recused* ourselves.

(7) We have *families* ourselves.

No speaker of English would be inclined to think of *families* (a noun) and *recused* (a verb) to have the same function, but here they are in the same sentential context: *We have X ourselves.*. The deception is due to the ambiguity of surface forms. In particular, forms that are common are most likely to occur with more than one structural category, as with *have* here. Extending the contexts to include standard Penn Treebank-style part-of-speech information may provide a richer structure, but in this case it is not helpful in distinguishing between the senses:

[1] The Penn Treebank (1992) contains no examples of *mosh* and only nominal senses of *text*.

(8) We_PRP have_VBP *recused_?* ourselves_PRP.

(9) We_PRP have_VBP *families_?* ourselves_PRP.

Therefore we must look deeper into the syntactic structure of the contexts, dealing with them not as surface regular expressions, but rich parse trees.

(10)

We	have	*recused*	ourselves	.
$\overline{NP}$	$(S[dcl]\backslash NP)/(S[pt]\backslash NP)$	$(S[pt]\backslash NP)/NP$	$\overline{NP}$	.
		$S[pt]\backslash NP$ (>)		
	$S[dcl]\backslash NP$ (>)			
$S[dcl]$ (<)				

(11)

We	have	*families*	ourselves	.
$\overline{NP}$	$(S[dcl]\backslash NP)/NP$	$\overline{NP}$	$(S\backslash NP)\backslash(S\backslash NP)$	.
	$S[dcl]\backslash NP$ (>)			
	$S[dcl]\backslash NP$ (<)			
$S[dcl]$ (<)				

These CCG derivations show that the different uses of the third word in each sentence are shown in the different category types given to the word ($(S[pt]\backslash NP)/NP$ or transitive participle for *recused* and NP or bare noun phrase for *families*) and also to the order of combination (*recused* combines first with *ourselves* and *families* with *have*.) It is apparent that the differences are due to the lexical ambiguity in the words to either side, and consequently to the internal structure of the parse tree.

Because both *have* and *ourselves* have (at least) two categorial senses, the structure of the sentence is dictated by the category of the target word. If that word is unknown, there is no way to distinguish between the two structures, and we must rely on prior probabilities to choose the most likely parse. If, however, the goal is to deduce the category of *have*, knowing whether the category of the third word is $(S[pt]\backslash NP)$ or NP will make it not only possible, but unambiguous.

1.2 Motivation from Engineering

A lexicon is composed of a set of mappings from words to labels. Various smoothing and backoff strategies are routinely used to make up for the fact that no lexicon covers the entirety of any natural language. These missing lexical items are problematic for all

computational parsing systems, especially if they cannot rely on a robust POS tagger to support backoff, but they are a major cause of error in strongly lexicalised parsers such as those based on CCG [Clark and Curran 2004; Hockenmaier 2003]. The problem is especially acute for languages for which there are fewer available resources for building NLP systems, but even in the case of the Penn Treebank for English, widely considered a large-scale corpus, we are aware of many CCG category types which it misses entirely [Clark et al. 2004]. This includes both words that do not occur even a single time in the corpus, such as *mosh* and words which do occur but for which some additional category is licensed, such as *text*.

The current method used by StatCCG [Hockenmaier 2003] for dealing with unseen words is part-of-speech backoff, which identifies words that are not in the lexicon, or have not been seen a sufficient number of times to have a reliable model, and replaces them with their POS tags. It affords an F-score of 93% over labelled dependencies in the optimal configuration, despite the fact that 18% of the correct lexical items (word/category pairs) are missing from the lexicon. It is difficult to predict how these affect dependency errors in parses, but when we examine category match accuracy of the StatCCG parser over §00 of CCGbank, we can attribute 80% of the error to the model and the beam, since those pairs are available to the parser. For the remaining 20%, the correct word/category pair is absent from the lexicon. Adding all the needed word/category pairs to the lexicon could result in up to 20% reduction in error, affecting nearly a third of the sentences in the test set. This is a very important area for systems that use parser output, as they necessitate wide coverage and accurate full-sentence parses. Further to reducing parsing error, a robust method for learning words from unlabelled data would result in the recovery of interesting and important category types that are missing from our standard lexical resources.

In this thesis we will address the problem of lexicon acquisition, which, by merit of operating within the lexicalised paradigm of CCG, is equivalent to grammar acquisition. The two major goals are to learn a statistical lexicon (a mapping between words and categories, weighted by expected frequency) and a language model (a dynamic distribution over head-dependencies.) Because of Zipf's Law, we anticipate that simply annotating more data will not be a practical method for widening parser coverage in the long term. Instead, we try to supplement a treebank lexicon by adding experience with unlabelled data.

1.3 Theory

How do humans learn a new word? We hear or read it in a context, from which we can deduce at least its syntactic role and possible subcategorisation frame, if not its semantics. We can look it up in a dictionary, or ask another speaker what it means. Often we will relate it to another word that we already know, linking it to a prototype that already exists in our lexicon. When an artificial system learns a new word, it has a different set of options. It cannot look it up in a dictionary, presumably because by definition any word that is unknown to the system is outside of its dictionary. It has a hard time asking someone else, but there are applications for that approach in co-training or active learning. In this thesis we will focus on the task of learning an unknown word's syntactic category based on the context in which it is seen.

Pullum and Scholz [2007] posit that there are too many exceptions to write an exhaustive learner based on intersubstitutability on the text level. One would need to go deeper to make syntactic generalisations, and the set of these is dependent on the grammar formalism of choice. In addition, the set of intersubstitutable categories is so fine-grained as to be prohibitive (as illustrated by the inexhaustive coverage of the Levin classes [Levin 1993]), so the relevant level of inquiry is not a word's single syntactic type, but a statistical distribution over a set of lexical categories.

Brighton et al. [2005]'s study of language evolution states that the most often-used words in a language have the freedom to be morphologically and syntactically irregular, because they are so often reinforced, but the rarer words have to conform to a system so that the whole lexicon can fit within human memory constraints. This suggests that, in the real task of learning the long tail, the job of finding the categories of previously unseen words would be simpler than learning all the words of a language from scratch, since we can assume that a word that has not been seen in a large amount of training data will tend to be rare enough that it has to be conform to a small set of regular patterns. Brighton et al. find that this is the case for morphological systems, but Steedman [2000] points out that there are rare words out there that, while regular, are used in ways that are not represented in the labelled data, such as *bet*, as in *I bet you five pounds you can't eat that whole pie*, which has the category $(((S\backslash NP)/NP)/NP)/S$, a verb subcategorising for three nominal arguments and one sentential. So on the one hand, the theory tells us that rare words should be easy to categorise, and in practice we find that they are not. There are even instances of whole category types missing from CCGbank, which are addressed in chapter 8 of this thesis.

There are several ways of framing the problem of learning words, all dependent on the framework and formalism in which the learner is working. A system could have an entry for a word/category pair, but not have accurate information about its distribution or use. It could have several valid categories for a word form, but still be missing one or more categories. Homonymy adds another level of complexity. Morphology and alternation classes can go a long way toward generalising an existing lexicon, and guiding how new forms are added, but when a word type is genuinely homonymous with another word type of a different category, the problem is serious. The system could know that *font* has the category *N*, but not understand that there are two types of fonts that occur in different contexts [2]. For a purely syntactic system, such as CCG, this distinction is moot: the category for both is *N*.

This thesis focuses on word-learning from context. It seeks to place previously unseen words into an existing lexicon with the correct category or categories, and to estimate the probability of the word/category pair to an extent that it is useful in subsequent parsing tasks. POS backoff and self-training based on it are used as comparative baselines.

1.4 Proposed Thesis

1.4.1 Hypothesis

This thesis hypothesises that a semi-supervised approach to lexicon acquisition can make use of the best parts of existing lexical resources and an abundance of unlabelled data, in order to produce an improved parser that is superior to the original without the expenditure of additional labelled resources. We propose that Chart Inference (CI) is an efficient and reliable strategy for deducing a ranked set of possible categories for an unknown word using the partial chart formed from the known words that surround it. CCG [Steedman 2000] is particularly suited to this problem, because category types have a direct relationship to the surrounding constituents. CI is designed to take advantage of the full power of a generative CCGbank-trained parser, with access to the full inventory of CCG combinators as well as non-combinatory rules from the trained model. We propose that CI is capable of learning category types that are completely missing from the lexicon, and of generating new lexical items that can be integrated into an existing parser to improve its coverage. We compare it to existing learning

[2] Typeface, fountain

systems designed for the same task in terms of both precision and efficiency.

CI is based on parsing a partial chart, where all but one of the categories for the terminals are known. That empty terminal cell is the target, for which CI produces an ordered list of possible categories, with probabilities estimated from a generative model. CI selects a distribution of root categories for each sentence based on its end punctuation, then performs a top-down modified chart parsing process which, given the parent and one sister, fills in the missing sister. It has access to the full model and rules of the original parser, plus the general CCG combinators in the form of inverted rules, as described in chapter 4.

The ideal lexicon should strike a balance between coverage and accuracy. Too specific, and the scores on the development set will outclass the scores on unseen data. Too general, and the parser will be prone to overgeneration, and suffer from an unnecessarily large search space. There are not established metrics for assessing the level of generality of a lexicon, and there is no benchmark with which to compare them. But we can approximate generality by measuring the lexical compression that the training process achieves as the number of word tokens or types, divided by the size of the resultant lexicon.

In StatCCG, POS-backoff is treated as a lexical problem. POS entries sit alongside lexical entries and share probability mass in the lexicon. This simplifies the backoff problem, but adds another layer of complexity to the generalisation problem. The decision to add a POS entry removes some of the probability mass from lexical entries of the same category, and creates a delicate balance at training time that supplemental learning processes must be careful not to disturb.

1.4.2 Challenges

There are two places to encode lexical behaviour in a generative CCG parser: in the lexicon and in the model. This problem can be seen clearly in an examination of the treatment of prepositional types. Traditional CCG would determine the only lexical necessity for a preposition to be PP/NP (*on* it), but prepositional phrases often occur as modifiers of nouns [$(NP\backslash NP)/NP$] (kids *on* drugs), predicates [$((S\backslash NP)\backslash(S\backslash NP))/NP$)] (walked *on* water) or sentences [$(S/S)/NP$ (*on* tuesday we swam) or $(S\backslash S)/NP$] (we swam *on* tuesday). Including all five of these category types for every preposition in the lexicon is not necessarily the most efficient use of resources. Putting them in the model as an unlexicalised rule [$PP/NP \rightarrow (NP\backslash NP)/NP$] allows for more generality

at the expense of widening the search space. However, it allows for a more compact lexicon that is capable of wider coverage. It is possible that because CCGbank has been semi-automatically translated from the Penn Treebank, rather than analysed afresh in a CCG framework, the parser does not represent the optimal balance of lexical items and rules.

The two best-performing publicly available CCG parsers, StatCCG Hockenmaier [2003] and the C&C Clark and Curran [2007] parser, are successful in part because they are trained on a relatively large number of reliably annotated sentences. In some cases the annotations are inconsistent or erroneous, but the parser is robust to these sources of noise. Self-training, which uses the parser itself to label new training data, breaks down because the sentences added are of inferior quality compared to those initial gold standard training sentences [Steedman et al. 2003b]. When the volume of less-reliably-generated training sentences begins to overwhelm the original gold-standard set, the quality of the lexicon decreases. In addition, even adding too many good entries can upset the balance of the lexical model as, for example, large numbers of new nouns can take some of the probability mass away from truly ambiguous lexical entries, so that their rarer senses have a better chance of being ranked above their nominal sense by the model. As such, the initial errors in the parser can get compounded by bad parses that use them. Strategies for dealing with this include marking the training sentences with a confidence value, where gold standard sentences are high and added parses can either have a low value or get graded according to learner confidence [Deoskar 2008; Deoskar et al. 2011].

1.5 Structure of Thesis

This thesis is organised in two halves: the first three chapters outline the background and theory behind the learning of words, and the remaining five comprise experimental results and analysis.

Chapter 2 sets out the background information necessary for understanding the rest of this thesis, including a description of the grammar formalism and linguistic resources used, as well as a review of the current literature relevant to the thesis as a whole. Chapters 3 and 4 formally set out the word learning systems evaluated in the subsequent chapters, the former representing the baseline systems and their implementations, and the latter comprising the novel methodological contribution of this thesis.

The remaining chapters test the word learning mechanisms in practice. They are ordered in increasing complexity of the corpora used and the learning tasks undertaken. Chapter 5 comprises the most basic tests of convergence and completeness, and deals with a toy corpus designed to test the relative strengths and weaknesses of the learning systems. Chapter 6 introduces a novel natural-language corpus which is also a contribution of this thesis. It tests the coverage of the learning systems in the presence of complete information, and also on a held-out set. Chapters 7 and 8 extend the scope to more conventional parsing tasks, using CCGbank first as a target set and then as a training set, seeking OOL items. The most comprehensive wide-coverage evaluation is undertaken in Chapter 9, where CCGbank is used as a training set and learning takes place over the English Gigaword corpus. A conclusion and discussion of future work follows.

A portion of this work was published as Thomforde and Steedman [2011].

Chapter 2

Background

2.1 Introduction

This chapter sets out the background necessary for understanding the main points of the thesis, including the basics of CCG and a discussion of related work. We also describe the software and data resources used in the remainder of the thesis. The chapter concludes with an estimate of the capacity of a word learner to improve upon the state-of-the-art results of the StatCCG parser.

2.2 Literature Review

Word learning is an important pursuit that has strands in a broad range of language tasks. The two most closely related fields to the work in this thesis are the acquisition of a grammar or lexicon from scratch, and the improvement of existing lexical resources. Either can be approached with any level of supervision, and for any language. Deep Lexical Acquisition is taken to be the discovery of an extensive usage grammar, encoding syntactic, morphological and semantic information about a set of words, as well as the set rules that govern it. Shallow acquisition ranges from clustering words according to their similarity, to acquiring subcategorisation frames for verbs. The shape of the approach is closely linked to the mechanics of the grammar formalism and the kind of information to be acquired. HPSG and CCG, for example, allow for the encoding of multiple levels of information in the lexicon, in a way that phrase structure grammars do not.

This section discusses a representative selection of work relevant to word learning. It varies along three major axes: the extent of supervision required, the depth

of linguistic information acquired, and the complexity of the initial lexical resources used. Within these environments, the learning targets are also varied: from syntactic category labels to subcategorisation frames to CCG category types to HPSG SYNSEM frames. We examine the related work in terms of the sources of linguistic information that inform their features, which ranges from n-grams and suffixes to packed charts, becoming more complex over the years as computational resources improve and grammar formalisms multiply.

2.2.1 Surface Forms

This research programme is inspired by the work of Brent [1993]. He worked with a system called *Lerner* that identified new verbs of six specified subcategorisation frames. Lerner first identified verbs in surface strings by their diathesis alternations, then recorded all the arguments in the vicinity of the verb, according to a hand-compiled list of stopwords that were deemed good indicators of lexical categories. Finally, hypothesis-testing was used to rule out all but the most reliable verb-frame pairings. The really novel part of this experiment was in seeding an initial lexicon with a small number of closed-class words that enabled Lerner to relatively unambiguously identify the relationship between words from their surface forms.

It was Brent who framed the problem of automatic lexical acquisition in terms of a human trying to learn a new language from just a grammar book and a large set of sentences of the language. Though this "ordinary grammar book" is purely theoretical and probably contains much more than just a lexicon, in practice Learner was only given a few stopwords and was kept within the bounds of English by a very selective learning algorithm. Though Brent attempted only a small part of the language, he showed that a large proportion of syntactic structure can be bootstrapped from a very small amount of linguistic information.

Clark undertook two relevant studies using distributional clustering to induce a lexicon. The first [Clark 2000] sought to learn categories by automatically clustering distributions of context trigrams, using Markov models on the text of the BNC. Once the contextual clusters are produced, this amounts to a tagging class for rare words, since membership in a category is determined by similarity to a cluster. Ambiguity is encoded as similarity to more than one cluster. The system must have the number of clusters pre-specified, which is a weakness. Clark [2001] extended this work by evaluating category membership based on mutual information of the words on either

side, using the Minimum Description Length (MDL) principle to restrain the size of the lexicon, rather than specifying the number of clusters.

Deane [2003] employed a variant of Latent Semantic Analysis on surface n-gram contexts to measure the similarity between words. His approach tried to find words that were semantically similar to known words, while not guaranteeing syntactic similarity. Deane compared words by their contexts and contexts by the words they license, showing that relatively rare constructions, such as ___ *his/her/their/your way home* are very specific about the verbs they allow in the gap, and that this information allows inductive learning of natural language. His investigations were limited to a small number of rare constructions, and typically focused on nouns, but are nonetheless encouraging in their success in the semantic domain.

Regier and Gahl [2004] contributed to the lexicon learning problem in the form of a mathematical proof-of-concept to provide counter-evidence for the poverty of stimulus argument. Using Bayes' rule and the "size principle," only a small number of positive examples were necessary for handling a set of sentences using the anaphoric *one*, because the learner must favour the hypothesis that is most likely to generate the positive examples, and not the incorrect ones (it knows they are incorrect because they are unseen). They showed that negative evidence is unnecessary for learning lexical/syntactic patterns, because the system can infer it from not having seen it.

2.2.2 Syntax

Following Brent, Manning [1993] sought to construct a subcategorisation dictionary for all verb types from unlabelled data. He first found arguments of target verbs using a finite-state parser over raw text, keeping count of the verbs and their subcategorisation frames in the process. He then filtered the results using hypothesis-testing along the same lines as Brent as discussed in chapter 1, but with a higher threshold, as the automatic parses were more errorful. As such, he succeeded in automating the process of finding unambiguous cues for all types of verbs, no longer limiting the process to those cues which can be manually constructed. He also condoned a high-precision, low-recall approach, stating that "the desire to combine hand-coded and automatically learned knowledge suggests that we should aim for a high-precision learner (even at some cost in coverage)..." [Manning 1993].

Also employing a parser for word learning, Briscoe and Carroll [1997] designed their system to acquire subcategorisation information for verbs, annotating contexts

with shallow parse structures and then automatically classifying the patterns into a dictionary of verb classes. This was followed by Korhonen [2002]'s refinement through statistical filtering of semantic frames. Carroll and Fang [2004] attempted to expand the coverage of the ERG lexicon by automatically acquiring verb subcategorisation frames. They processed their email corpus with the RASP tools in order to provide rich morphological and syntactic features from which to extract frames. Combining the resources of a precision grammar with unsupervised learning over an unlabelled corpus, they were able to improve lexical coverage from 85.9 to 94.4%. Preiss et al. [2007] built on this approach by developing a rule-based system for learning subcategorisation frames for open-class words. They specified a level of grammatical relations over head dependencies and extracted frames from these rather than from parse trees, which allowed for abstraction away from any particular grammar formalism. Buttery and Korhonen [2007] have extended that system to a corpus of child-directed speech.

Zhang et al. [2010]'s task was to classify a very narrow set of non-compositional verb-particle constructions into four categories using a precision grammar. More interesting than their classifier was their choice of features, which were based on the partial chart generated by their parser when the target word was replaced by a dummy. They found that performance was correlated with the frequency of the target word in the training corpus; although the features only encoded partial information, many data points could make up a fuller picture of a word's properties.

Although these systems focus on learning verbs, the problem of word learning is much wider, and a comprehensive system must cover all lexical types. One approach is the clustering of words into sets according to their distributional similarity. Lin [1998], for example, used a dependency-based feature set to automatically learn a WordNet-like thesaurus. He trained his system on automatically parsed text containing dependency and part-of-speech annotation. He then constructed vectors for each word type that contained the dependencies they were likely to co-occur with, and clustered these according to similarity. Lin's system was able to learn nouns and adjectives fairly well, but struggled with verbs because their syntactic contexts were far more varied with respect to argument structure.

In a similar task to Lin's, Widdows [2003] attempted to automatically place new words into an existing taxonomy by learning clusters of structural features from unlabelled data. This also qualifies as a semi-supervised method, as the clusters were partially known, having been seeded with parts of WordNet. Widdows created feature sets of selected words in a predefined context window, reduced the dimensionality with

Latent Semantic Analysis, and then clustered them according to the cosine similarity of the reduced feature vectors. Using the classification method, he was able to place common nouns, proper nouns, and verbs into the taxonomy.

Watkinson and Manandhar [1999a] produced the lexicon learning system most closely relevant to this thesis. Their learner made use of a statistical parser to choose the best candidate label for an unknown from the set of all known categories. Their concern was with a compact lexicon, so they reexamined past sentences each time they made a change to the lexicon, in order to produce a set of lexical items that worked most efficiently together. They highlight the fact that working within the lexicalised formalism of CCG equates learning the lexicon with learning the grammar. A version of Watkinson's system is used as a baseline in this thesis and will be discussed further in chapter 3.

A related field of investigation is the automatic discovery of whole grammars, which shares our goal of learning the rules and lexicon of a language, but attempts to learn the whole grammar at once. Osborne and Briscoe [1997] implemented one such system, which learned an unsupervised stochastic categorial grammar using a combination of EM and MDL over bracketed parts of speech. Klein and Manning [2005] also worked in the unsupervised domain, using EM and distributional clustering to induce a model of linguistic structure from WSJ-10.

2.2.3 Morphology

While the features used by these systems are indeed rich and informative, the range of available linguistic information extends beyond the syntactic. Baldwin et al. [2005] notes that deep lexical acquisition can be informed by morphology, syntax, ontology, or any combination of these features, and the same holds true for shallower systems. For example, Cucerzan and Yarowsky [2000] used raw text to learn new words on a morphological level, predicting the categories of unseen words from the distributional similarity of their suffixes to those of known words. They also take a hybrid approach with their data, using the small amount of annotated text for training the vectors, then mining the unlabelled data for new words to compare to the classification scheme learned in the first phase. Not only did the unlabelled text help by providing evidence for the distributional patterns of unseen words, it also boosted the counts for the known words, such that the whole process is motivated by bare text, and enabled by a small seed set of word-category pairs.

Their focus on minimal supervision is a contentious one. Ideally, we should be aiming for maximum supervision, fully exploiting the combined power of all the rare but reliable labelled data and human effort that are available, while enabling our systems to take full advantage of the abundance of cheap unannotated data. Though Cucerzan and Yarowsky [2000] is useful in showing that small amounts of supervision are sufficient, it suggests that hybrid methods are indeed the future.

2.2.4 Semantics

Semantic information has also been exploited for lexical acquisition tasks. Grenager and Manning [2006] set out to learn subcategorisation frames from unlabelled data using EM to map a set of syntactically derivable arguments to their semantic roles. Their training data consisted of syntactic parse trees with no semantic annotation. By defining a small set of features that depend only on the syntactic and morphological structure of the text, they were able to move up a level to learning the semantic roles of the arguments for each verb, which equates to subcategorisation information. They made prudent use of unlabelled data by defining a set of features, gathering all the feature/target data pairs, and applying EM to find the mappings that gave the best results. This process was designed for verbs, but in theory it could be expanded to learn any relation between a constituent and its semantic role, provided the right set of features were at hand. It is worth noting that although the learning process itself is unsupervised, Grenager and Manning did peek at the gold standard to apply the correct labels to the learned clusters. This is only a very small intrusion into the supervision, but it does betray the fact that designing a system that learns automatically from unlabelled data will not necessarily come up with answers that fit neatly into human-designed language formalisms.

Zettlemoyer and Collins [2005] also used semantic information, inducing a CCG grammar from pairs of database queries and their lambda calculus representations. Syntax was therefore learned as a byproduct of mapping surface forms to lexical entries through their GENLEX learning algorithm. Though the training data was not annotated with parse trees, they were able to ensure precision by concentrating on finding a small lexicon that would result in the correct logical forms for each surface string. In addition to the labelled sentences, the input to their learning algorithm included an initial lexicon seeded with a variety of entries, including place names, common constructions, and stopwords, which they deemed “easily specified by hand.” It is also

worth noting that their GENLEX algorithm generated all the possible mappings from surface to logical forms, and therefore produces many spurious lexical entries, which were then eliminated in a later pruning stage.

HPSG lends itself to the combination of syntactic and semantic features. Fouvry [2003] took a very basic approach to deep lexical acquisition, assigning a generic underspecified frame to all unknown words. Bits of lexical information were slotted into these frames as and when the morphological or contextual cues supported them. His work set a precedent echoed by many subsequent systems, including Cholakov and van Noord [2010a], who again treated lexical entries as collections of paradigms. They simulated unknownness in a Dutch HPSG Treebank by removing a set of words from their training set, then attempting to recover the most common 95% of them from a combination of morphology and context. Though their best performance was on nouns, they were able to reduce their error rate by 4%, while keeping parsing times level. They note that it was not uncommon for their system to learn correct bits of information, but get penalised for them at evaluation time because they were missing from the gold-standard lexicon. Barg and Walther [1998]'s approach to lexicon acquisition took a similar shape, in that they did not treat words as simply known or unknown, but subject to change in an online learning context. Working with a canonical fragment of HPSG for German, they acquired bits of SYNSEM information in word paradigms using an update and unification mechanism that allowed features to be missing, incomplete or in conflict, and resolved only when enough supporting context was encountered. Their main problem was the automation of the decision to halt learning for individual words; they note that "[i]t is a topic of future research when to consider information certain and when to make revisable information restrictive."

2.2.5 Further Afield

Much recent work in the acquisition of lexicons and grammars has used a combination of the feature types discussed above, and incorporated sources of linguistic information derived in other fields. Baldwin [2005] attempted to improve an existing lexicon using deep lexical features based on a combination of morphology, syntax and ontology. He found the syntax-based learning most effective for learning the open classes of words, morphological features helpful only for nouns, and ontological information from WordNet both difficult to execute and below the baseline for most word types. This was followed up by a system that used conditional random fields over a range

of lexical and contextual features to generate supertags for word tokens [Blunsom and Baldwin 2006]. Zettlemoyer and Collins [2005] seeded their system with stopwords and learned the grammar rules as a byproduct of mapping surface strings to logical forms. Deoskar [2009] took a sophisticated approach to the induction of PCFG parsing models, using unlabelled data and smoothing to learn accurate estimates of lexical parameters, and moved on to semi-supervised learning to focus on the rich Zipfian tail of unknown words [Deoskar and Rooth 2008; Deoskar et al. 2009; 2011]. Snyder et al. [2009] and Snyder and Barzilay [2010] made use of the information in parallel structures of bilingual data to improve their parsing models. Kwiatkowski et al. [2010; 2011] took advantage of the structural relationship between syntax and semantics to drive grammar induction on both child-directed and standard query data.

Error mining represents another step up in learning complexity. It not only uses a parser to enrich the features about syntactic behaviour, but leverages the feedback from the parser as another feature dimension to indicate where more learning is required. van Noord [2004]'s work with error mining of an HPSG treebank can be considered a form of grammar learning starting with a very large initial lexical endowment. He achieved a 4% increase in coverage over the original grammar by analysing the n-grams in the parser output in terms of parseablility. The n-grams that went most commonly un-parsed were considered points of deficiency and presented in a ranked table to a human expert for repair, either in the lexicon or the original treebank. This approach suffers from the fact that it can only diagnose non-parse errors, and can't find the errors in the lexicon that cause incorrect parses to be produced, which are more problematic to the end user. Cholakov and van Noord [2010b] built a system for identifying words that participated in parsing errors and improving their lexical entries. They generated specific training sets for each target word by composing an internet search, and finding the frames that cover 80% of the results. The target word, though known with respect to the initial lexicon, was temporarily treated as unknown and re-acquired with the larger training set. They tested it with only 36 target unigrams, but had promising results. This is an indicative example of the value of data volume in distinguishing between rare constructions and noise.

Rule-based lexical acquisition is an unpopular but relevant field. One such effort was Cussens and Pulman [2000], which attempted to learn HPSG grammar rules using linguistic constraints coded in Prolog. Given a complete rule system, they delete one or more rules from the grammar and attempt to recover them, based on maximising features from the partial parse chart. The solution is presented in terms of ranked set

of hypotheses for examination by a human annotator. They remark that their system is designed to acquire rules rather than lexical items, but that other systems may operate on the lexical level by treating lexical entries as a type of grammar rule. As this distinction is easily blurred in CCG, we take advantage of their suggestion. A recent lexical acquisition system, uniquely employing a small set of CG-based rules, was that of Yao et al. [2009a;b]. Though their system was small, it derived the category of unknown words using simple transformation operations on the surrounding categories, yielding a fast system that was capable of learning more linguistic patterns than were licensed by the original lexicon. This system has a strong bearing on ours, and we implement a statistical extension of it as a baseline. Both the paper and the learner will be discussed further in chapter 3.

2.2.6 Analysis

From this body of work we can abstract several recurring principles. First, context, in its various forms, has a powerful contribution to make to word learning. Whether one looks at bare words in a large context window, or well-defined features based on position or dependency, there is information to be gained from contexts. The trick is getting that information to translate to a useful classification of the target words. For CCG, it entails mapping the location and type of surrounding constituents directly to the form of the category type. The learning step is the most widely varied and the most unexplored. The use in related work of vector-clustering and pair matching with EM, LSA, and so on, represents a trend of applying well-proven machine learning methods to hard-earned data, a sound plan in theory, but we contend that more can be done to explore learning methods that are specific to language.

Additionally, we see a common thread in deciding what amount of supervised linguistic guidance is appropriate. Several of the systems discussed above were provided with small, highly accurate lists of stopwords: where Brent used closed-class words, Zettlemoyer and Collins extended that to include proper nouns and question words. Grenager and Manning peeked at the gold standard to apply labels to their learned classes, and Cucerzan and Yarowsky used the output from the supervised phase to constrain the form of the unsupervised. Both Widdows and Lin slotted their learned words into a pre-existing taxonomy. They all show that parametric linguistic information is enormously beneficial to the final output of the system, giving it form that would otherwise be very difficult for a completely unsupervised system to learn. This

suggests that we should be investing more effort into ensuring that our small amount of hard-won linguistic information is highly reliable, then bootstrapping the rest of the grammar from unlabelled data.

Since Brent, as computational power increases and lexical resources become both deeper and wider, the sources of lexical information used to learn words and grammars has increased in both complexity and creativity. What began with surface forms quickly evolved into using morphological information, then a parser, then a range of semantic, ontological, and contextual features, the complexity of which require large-scale computational resources that we are only now able to access. The future of word learning will take the best parts of these and use them in combination with the best machine learning techniques of the day, in order to learn a grammar more quickly and more precisely than any of these individual methods.

The word-learning system introduced by this thesis is rooted firmly in the syntactic tradition, using a parser to enrich the information in unlabelled surface forms, and also to build a partial chart to enable the use of structural contextual information. The result is a semi-supervised word learning system for acquiring syntactic categories for unknown words in several applications along a spectrum from lexicon induction and bootstrapping to domain adaptation of a wide-coverage parser. Where Cucerzan and Yarowsky [2000] extoll the virtues of minimal supervision, we embrace maximal supervision, insofar as we use as much of the power of the labelled data as possible, while incorporating anything that can be reliably learned from unlabelled data. In addition, learning a specific class of words (like verbs) is a very different task to learning the categories of all kinds of words. A successful lexicon learner should be able to handle all possible categories in the language, and should be able to invent new categories when its inventory is insufficient to describe rare behaviours.

2.3 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) [Steedman 2000] is a fully lexicalised descendent of Categorial Grammar (CG) [Bar-Hillel 1953] that is especially powerful in expressing long-distance dependencies. It stands apart from other theories in that it pairs lexical constituents with category names that express their structural role within the derivation. CCG is the formalism of choice for a number of parsing efforts. The most widely used is the C&C parser of Clark and Curran [2003; 2006; 2007], which employs a log-linear model as well as a supertagger [Bangalore and Joshi 1999] to

report state-of-the-art results with field-leading efficiency.

Hockenmaier [2003]'s StatCCG parser is not as fast, but has the benefit of being largely modular, as well as generative. Both are trained on CCGbank [Hockenmaier and Steedman 2007], a semi-automatically translated CCG version of the Penn Treebank. OpenCCG [White and Baldridge 2003] and the statistical extension StatOpenCCG [Christodoulopoulos 2008] are available to the public and very adaptable, though they are not commonly used in state-of-the art applications. Also competitive in the field is Doran and Srinivas [1997], who trained their CCG parser on a small XTAG corpus and achieved state-of-the-art results with the use of a supertagger. CCG grammars are being constructed for a number of other languages as well, including Turkish [Cakici 2005; Bozsahin 2002], German [Hockenmaier 2006], Thai [Jaruskulchai 1998], and Japanese [Otani and Steedman 2007].

2.3.1 Categories

The *category* is the central unit of linguistic information in CCG. Unlike in other lexicalised formalisms, where the names of the categories are used as a shortcut or a lookup for a word's behaviour, in CCG the category names themselves encode the word's relationships to the other constituents in the sentence.

Atomic categories are the building blocks of the lexicon. We will treat them as having a *valency* of 0, since they have only a result, and no arguments. Any symbol could be used for an atomic category, but the canonical set in CCGbank is limited to S, N, NP, PP, $conj$, and a number of types reserved for punctuation. Any of these may be modified with a range of bracketed type options, such as the atomic category below, which has a type of $[dcl]$, indicating a declarative sentence.

An atomic category: $S[dcl]$

A complex category: $(\underbrace{S[dcl]}_{\text{Result}} \backslash \underbrace{NP}_{\text{Arg 2}})/\underbrace{(S[to]\backslash NP)}_{\text{Arg 1}}$

Complex categories are created by joining atomic categories with a slash to the right or left. The complex category above has a *result* of S, and two *arguments*, one of which is itself complex. This category has a valency of 2, since it is composed of a result and two arguments. For the purposes of this thesis, we number the arguments from right to left, indicating their depth, which is related to the proximity of the constituents they combine with. Arg 1 above, appearing after a rightward slash, is considered a

Forward Application	A/B	B	$\rightarrow$	A	$(>)$
Backward Application	B	$A\backslash B$	$\rightarrow$	A	$(<)$
Forward Composition	A/C	C/B	$\rightarrow$	A/B	$(>B)$
Backward Composition	$C\backslash B$	$A\backslash C$	$\rightarrow$	$A\backslash B$	$(<B)$
Forward Crossed Composition	A/C	$C\backslash B$	$\rightarrow$	$A\backslash B$	$(>B_x)$
Backward Crossed Composition	C/B	$A\backslash C$	$\rightarrow$	A/B	$(<B_x)$
Forward Substitution	$(A/C)/B$	C/B	$\rightarrow$	A/B	$(>S)$
Backward Substitution	$C\backslash B$	$(A\backslash C)\backslash B$	$\rightarrow$	$A\backslash B$	$(<S)$
Type-raising	X		$\rightarrow$	$T/(T\backslash X)$	$(>T)$

Figure 2.1: Inventory of CCG combinators.

rightward argument, and Arg 2 leftward. In this thesis, and in CCGbank, parentheses are always used to avoid confusion[1]. We will refer to both the result and argument parts of a category type as *categorial elements*. The upper limit on valency is generally accepted as 4, but in practice we find that some rare categories exceed this limit. An insightful reanalysis of the annotation of the treebank may be able to reformulate the lexicon to satisfy this valency restriction.

2.3.2 Combinators

Combinators are CCG's standard mechanism for combining constituents to produce a result, and are one type of *rule* used in CCG parsing. Figure 2.1 shows the standard inventory of CCG combinators in CCGbank. The first four combinators (application and composition) dominate the treebank. The next four (crossed composition and substitution) are less common. The type-raising rule is used in a limited set of cases, and we have to be careful that it is not over-applied [Hockenmaier 2003]. In this thesis we ignore the more sophisticated multi-modal extension to CCG by Baldridge and Kruijff [2003], which limits overgeneration by restricting the use of certain rules to only appropriate situations, but only because the parsers we are working with do not support them. The word-learning systems described herein would benefit from modality restrictions on the use of many of the combinators, so as to avoid spurious inferences.

Type-raising is the only unary rule specified by classical CCG, but it is used in very limited circumstances. In fact, subject type-raising, the most common use of

[1]CCG specifies a left-to-right reading, so $S\backslash NP/NP$ is read as $(S\backslash NP)/NP$ and not $S\backslash(NP/NP)$

this rule [Steedman 1991], accounts for only 2% of all unary rule instances present in CCGbank[2]. Theory dictates that the only two components of a CCG grammar are the lexicon and the combinators, but in practice, we need a number of non-combinatory *rules*, or transformations, to encode the practical grammar of a natural language. This means that CCG may conflate grammar induction with lexicon induction on paper, but in reality there is a lot more to be learned in order to produce a competent parser capable of all the intricacies of natural language.

2.3.3 Derivations

A *derivation* applies combinators to categories and yields an interpretation for a sentence. Figure 2.2 gives an example CCG derivation for a simple sentence. Each token in the sentence is first underscored with a category type. Combination is signified by a horizontal line indicating the scope of the rule, and labelled with the rule type used. The result category is written under the line.

The final rule used in this derivation is the non-combinatory convention of attaching a full stop to the end of a complete sentence, labeled as (M) to denote a *model rule*, one that is not motivated by any of the standard CCG combinators, but made available to the parser through the lexicalised rules that are integrated with the parsing model. The canonical formal description of CCG would suggest that the full stop be given the category $S\backslash S$, but the treatment of individual lexical entries and their behaviour concerning model rules is the choice of the individual parser, and can be much freer than prescribed formal linguistic theory, in order to cope with the vagaries of the corpus and the annotation scheme used.

2.4 Parsing with CCG

There are three major CCG parsers available to the public: the generative StatCCG, the discriminative C&C parser, and the mostly-for-educational-purposes OpenCCG. In this thesis we utilise the resources of StatCCG and its lexical source material CCGbank for training and evaluation, and StatOpenCCG, a statistical version of OpenCCG, for the learning phase. This section specifies the extent and nature of these resources.

[2]Figure estimated from 2486 instances of $NP \rightarrow S/(S\backslash NP)$ and 132809 unary transformations in total, from derivations in CCGbank §02-21.

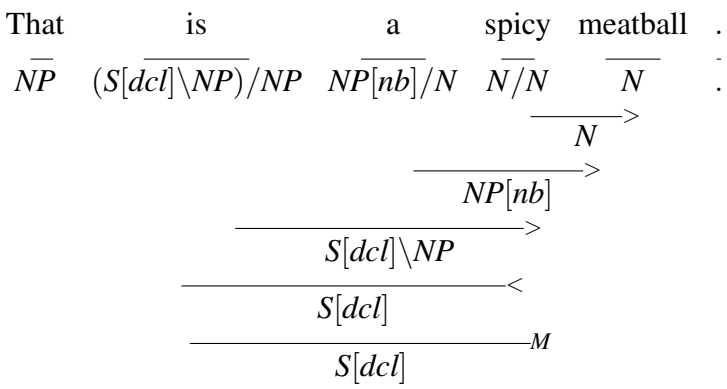


Figure 2.2: A sample CCG derivation.

2.4.1 StatCCG and CCGbank

StatCCG [Hockenmaier 2003] is a generative parser trained on CCGbank [Hockenmaier and Steedman 2007], a CCG translation of the Penn Treebank. It parses POS-tagged input files that have one sentence per line. In its optimum configuration, it achieves an F-score of 93% over dependencies on §00 when trained on §02-21. CCGbank is delivered in a machine-readable format annotated with POS, category, and dependency information specifying a full CCG parse tree.

2.4.1.1 Lexicon

Figure 2.3 shows a snippet of the StatCCG lexicon, reflecting the structure of the files. The lexicon is generative, representing the probability of each word (W) given its category (C): $P(W|C)$. As such, the probabilities do not sum to 1 over all words, for example, the word *framing*, but rather over all words within a category C, representing, for example, the word types that have the categories N and $((S[ng]\backslash NP)/PP)/NP$. This property of the generative model leads to a bias for rare category types to be chosen, which is not a problem when the parser is being used to create a derivation for a sentence, but does manifest when we are using that parser to generate and weight category types for unseen items, which are more likely to be open-class words.

StatCCG's generalisation mechanism is a word frequency threshold, which dictates how often a word type must be seen in the training corpus to justify its inclusion in the lexicon. If the word is seen N or more times, it enters the lexicon paired with all the categories it has been seen with. If it has not been seen a sufficient number of times, it is replaced by its POS tag in every instance, and the distribution over categories for the POS tag calculated accordingly. The POS entry is then consulted when an OOL

W	C	$P(W\|C)$
frame	N	3.39291946178603e-05
frame	$(S[b]\backslash NP)/NP$	0.000153456610143482
framed	$S[pss]\backslash NP$	0.000286204922724671
framed	$(S[pss]\backslash NP)/PP$	0.000425170068027211
framed	$(S[pt]\backslash NP)/NP$	0.000425531914893617
framers	N	9.69405560510295e-06
frames	N	1.45410834076544e-05
framework	N	4.84702780255148e-06
Framework	N	4.84702780255148e-06
framing	N	4.84702780255148e-06
framing	$((S[ng]\backslash NP)/PP)/NP$	0.00141843971631206
Framingham	$(N/N)/(N/N)$	0.000171526586620926

Figure 2.3: A snippet of the StatCCG lexicon, trained on CCGbank §02-21.

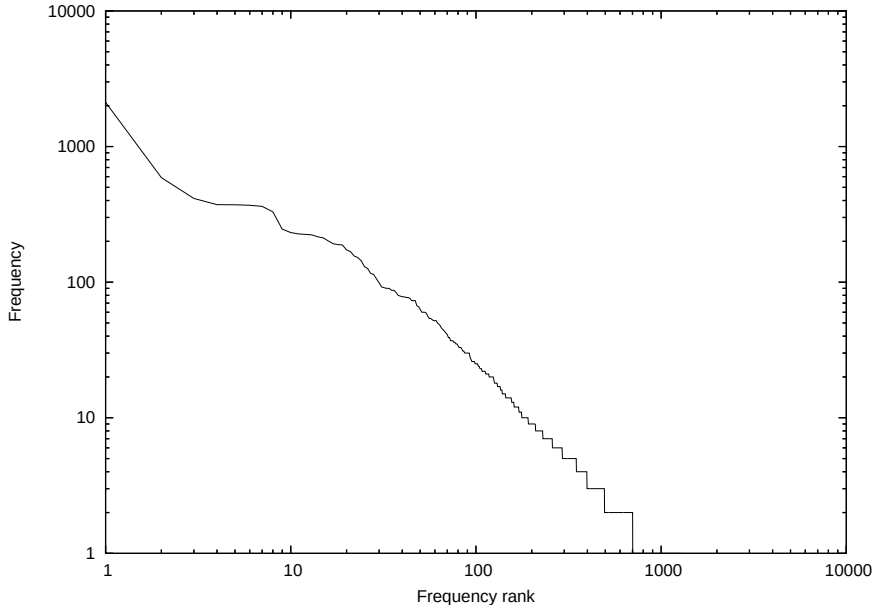


Figure 2.4: Zipfian Rank v. Frequency distribution of $W|C$ pairs in CCGbank.

word is seen at test time. Setting this word frequency threshold determines the level of generality of the lexicon by dictating how much of the training data will be turned over to POS-entries, and by extension, how often POS-backoff will be relied upon.

Figure 2.4 shows the expected Zipfian distribution of word/category pairs in §02-21 of CCGbank. It indicates that the majority of lexical items are seen fewer than ten times in the corpus. This means that the problem of the long tail applies not only to completely unseen items, but also to rarely seen items, which may be folded into the POS model depending on the word frequency threshold.

2.4.1.2 Model

In theory, all that a CCG grammar consists of is a lexicon that maps words to categories and a set of combinators. Statistical parsers also require some model with which to rank the parses they generate. In practice, however, Hockenmaier and Steedman [2002] have found it necessary to model a large number of transformations, both unary and binary, that are found in the treebank, but are not strictly covered by either the lexicon or the combinators. They have also implemented special treatment for coordination and punctuation. In addition to the lexical model of $P(W|C)$, StatCCG contains a word-word head dependency model with which the probabilities of derivations are calculated. The components of the StatCCG head-dependency model are:

Expansion probability $P(exp|C)$, $exp \in left, right, unary, leaf$

Head probability $P(H|C,exp)$

Non-head probability $P(D|C,exp,H)$

Lexical probability $P(w|C,exp = leaf)$

2.4.1.3 Rules

In theory a CCG grammar needs only the standard combinators listed in figure 2.1, but in practise the grammar of StatCCG is made of a set of lexicalised transformation rules. There are 200 distinct lexical rules present in CCGbank, the overwhelming majority of which are unary rules, most of which are used only a handful of times. These are derived from all the rule instantiations in the Penn Treebank. The rules are not encoded separately in StatCCG, but are in StatOpenCCG, so we can examine them singly. Since

$$
\begin{array}{lcl}
NP & \rightarrow & NP/(NP\backslash NP) \\
NP & \rightarrow & ((S\backslash NP)/(S[adj]\backslash NP))\backslash(((S\backslash NP)/(S[adj]\backslash NP))/NP) \\
NP & \rightarrow & ((S\backslash NP)/(S[b]\backslash NP))\backslash(((S\backslash NP)/(S[b]\backslash NP))/NP) \\
NP & \rightarrow & (S\backslash NP)\backslash((S\backslash NP)/NP) \\
NP & \rightarrow & ((S\backslash NP)/(S\backslash NP))\backslash(((S\backslash NP)/(S\backslash NP))/NP) \\
NP & \rightarrow & (S/(S[b]\backslash NP))\backslash((S/(S[b]\backslash NP))/NP) \\
NP & \rightarrow & S/(S/NP) \\
NP & \rightarrow & S/(S\backslash NP) \\
NP & \rightarrow & S\backslash(S/NP) \\
NP & \rightarrow & (S/(S[pss]\backslash NP))\backslash((S/(S[pss]\backslash NP))/NP) \\
NP & \rightarrow & (S/(S[to]\backslash NP))\backslash((S/(S[to]\backslash NP))/NP)
\end{array}
$$

Figure 2.5: Inventory of unary rules in CCGbank that expand from NP.

there is no probability associated with each combinator[3], they are often over-applied in the parsing step, especially by StatOpenCCG.

Figure 2.5 lists a subset of the unary rules present in CCGbank. While some seem necessary, like $NP \rightarrow NP/(NP\backslash NP)$, which is used for NP modification, others may arise from annotation errors or vagaries of the translation to CCG. These rare or erroneous rules incur no penalty for their use, since the parsing model does not take the probability of the rule into account, and so they are a potential source of overgeneration error, especially when parsing corpora of a different domain.

Coordination is treated with a special ternary operation that is internal to the parser, and triggered by the category type $conj$. Figure 2.6 shows the distribution of the category $conj$. Any constituent can be coordinated, and take on an additional type $[conj]$, even if it already has a type. This gives rise to category types in internal parse nodes like $S[em][conj]$ or $(S\backslash NP)\backslash(S\backslash NP)[conj]$.

While all special symbols are treated like lexical items, with a full CCG category that encodes their function (for example $\$: N/N[num]$), the punctuation symbols receive special treatment. Figure 2.7 shows the special punctuation categories. All end-punctuation symbols are given the category (.), or the full stop. Sentence-internal punctuation has three possible category labels: {: ; ,}. These punctuation categories are in rare cases given erroneously to other word types; figure 2.8 is just one indica-

[3]Both parsers get their expansion probabilities from the head-dependency model, which represents the input and output categories, but not the specific model rule or combinator used to combine them.

W	C	$p(W\|C)$
and	$conj$	0.785969261279127
or	$conj$	0.119286068418443
but	$conj$	0.0772930094199306
than	$conj$	0.00282597917699554
nor	$conj$	0.00188398611799703
plus	$conj$	0.0014873574615766
of	$conj$	0.00118988596926128
as	$conj$	0.0060485870104115
AND	$conj$	0.000644521566683193
if	$conj$	0.000545364402578086
yet	$conj$	0.000347050074367873
And	$conj$	0.00029747149231532
both	$conj$	0.000198314328210213
'n'	$conj$	0.000198314328210213
so	$conj$	0.000198314328210213
But	$conj$	0.00014873574615766
&	$conj$	0.00014873574615766
either	$conj$	0.00014873574615766
not	$conj$	0.00014873574615766
only	$conj$	0.00014873574615766
vs.	$conj$	0.00014873574615766
just	$conj$	9.91571641051066e-05
minus	$conj$	9.91571641051066e-05
with	$conj$	9.91571641051066e-05
andor	$conj$	4.95785820525533e-05
et	$conj$	4.95785820525533e-05
'N	$conj$	4.95785820525533e-05
neither	$conj$	4.95785820525533e-05
Or	$conj$	4.95785820525533e-05
then	$conj$	4.95785820525533e-05
v.	$conj$	4.95785820525533e-05
versus	$conj$	4.95785820525533e-05

Figure 2.6: Conjunction category distribution in CCGbank.

W	C	$p(W\|C)$
!	.	0.0015796178343949
’	:	0.00161812297734628
?	.	0.00998726114649682
-	:	0.0226537216828479
:	:	0.273462783171521
.	.	0.988433121019108
,	,	0.999917208262615
;	;	1
...	:	0.10302049622438
–	:	0.598705501618123
-RRB-	:	0.000539374325782093

Figure 2.7: Specialised punctuation categories in CCGbank.

W	C	$p(W\|C)$
,	,	0.999917208262615
2	,	2.06979343461523e-05
an	,	2.06979343461523e-05
section	,	2.06979343461523e-05
underwriters	,	2.06979343461523e-05

Figure 2.8: Distribution of the comma category in CCGbank, including lexicon errors.

tion of the scale of tiny errors in CCGbank, the correction of which represents fruitful future work.

2.4.1.4 Backoff

Part-of-speech backoff is the general method used for dealing with unseen words and accounts for a large degree of the success of StatCCG. When StatCCG is presented with an OOL word, it simply backs off to the lexicon entry for its part-of-speech instead. This of course requires that the input data is POS-tagged, which is a potential source of error. Also, there is often a much wider scope of category ambiguity imparted by the POS-entry than the lexical entry. This makes the parsing process longer and more vulnerable to misparses.

The way the model is constructed, a file of common words must be provided to the parser at training time. The current best threshold, as described by Hockenmaier and Steedman [2007] is 30. This means that any word type that has occurred 30 or more times in the training sentences will have its word/category pair entered in the lexicon. Any word that is rarer than that threshold will have its POS/category pair entered instead. This is a clever trick that means the model contains separate information about common words and rare words. It is beneficial in theory because common words tend to be the irregular ones and rare words the regular, after Brighton et al. [2005], so their distributions will naturally be disparate. However, its downside is that much of the probability mass that could be used for known words is taken up by the POS-backoff entries.

Throughout this thesis we use different configurations for the frequency threshold. We endeavour to use the most appropriate value depending on the size of the training corpus. While 30 may be optimal for a parser trained on 40,000 sentences, when we add a lot more sentences to that, the parameter should rise, and when we use smaller training sets, it should fall.

2.4.1.5 Evaluation

Two forms of evaluation are relevant to our experiments, depending on the availability of annotation on the test set: head dependencies and category labels. StatCCG output is canonically evaluated on dependencies [Clark et al. 2002]. Hockenmaier makes the case that this is the fairest way to compare parses produced by CCG with those of other grammar formalisms, owing to the differences in the format of CCG being

incompatible with the standard evaluation mechanisms. However, for comparing one CCG lexicon to another, it is easier and more theoretically sound to use the category assignments at leaf nodes. This follows from the notion that assigning categories to the string of words is the hardest part of CCG parsing, and forming the rest of the tree is straightforward, as it follows the normal form principle [Hockenmaier 2003]. Throughout this thesis this metric will be referred to as *Category Match Accuracy*.

$$Precision = \frac{tp}{tp+fp} \tag{2.1}$$

$$Recall = \frac{tp}{tp+fn} \tag{2.2}$$

$$F1 = \frac{2PR}{(P+R)} \tag{2.3}$$

We calculate precision (2.1) as the number of true positives over the sum of true and false positives. Recall (2.2) is true positives over the sum of true positives and false negatives. F-score (2.3) is the harmonic mean of the two, equally weighted. We employ this standard method for calculating recall, even though Hockenmaier's slightly nonstandard calculation yields higher reported figures. Where Hockenmaier divides over the number of dependencies in the reference set that are comparable to the test set, we divide over the total number of dependencies in the reference set. In order to ensure commensurability when comparing our system to a StatCCG baseline, we have run the StatCCG parser ourselves and calculated precision and recall figures according to this scheme, rather than reporting published figures.

2.4.2 StatOpenCCG

StatOpenCCG [Christodoulopoulos 2008] is a statistical extension of OpenCCG [White and Baldridge 2003]. It employs the same model as StatCCG, with a few exceptions. We use it in place of StatCCG in the learning phase of the experiments in this thesis because it has a number of extra parameters that allow it to produce partial parses, run with POS backoff disabled, and accept lexicons independent of the model.

We train StatOpenCCG on the same CCGbank-style derivation files as StatCCG. It employs the same head-dependency model and calculates parse probabilities in the same manner. Coordination is treated somewhat differently, as two binary levels of combination rather than a single special ternary rule.

StatOpenCCG is used only in the training phases of the experiments herein. It is

not quite as good as StatCCG in parsing §00, but the several advantages offered by the expanded options make it ideal for the situations in which it is used.

2.4.3 The C&C Parser

In contrast to the two generative CCG parsers mentioned above, the C&C parser [Clark and Curran 2003] is a lexicalised discriminative parser which is trained on CCGbank and equipped with a supertagger, making it the fastest CCG parser to date. They report labelled dependency results of 86.6% precision and 86.3% recall, with category match accuracy of 93.6. In addition, the C&C parser has been used in several state-of-the-art training and domain-adaptation experiments. Clark and Curran [2006] show that training the parser on just sequences of categories, as suggested by Hwa [1999], rather than full trees, only results in a 1.3-point decrease in F-score. They note that

> ...the significant amount of syntactic information in CCG lexical categories allows us to infer attachment information in many cases. [Clark and Curran 2006]

The C&C parser was not used in this thesis, due to its discriminative model and the relative inaccessibility of the elements of the code. However, its speed and accuracy must be taken into account when evaluating our results, and future work would do well to re-implement our work in discriminative terms.

2.5 Room for Improvement

Out-of-Lexicon (OOL) words contribute to a specific number of errors for the parser, although it is difficult to exclusively assign blame amongst the many sources of error. Various parsers attribute between 15% and 89% or all parse errors to OOL words, depending on their domain and formalism, cf. Szolovits [2003]; Yona and Wintner [2008]; Fouvry [2003]. We attempt to calculate the amount of error caused by OOL words by checking the categories in the output parses against the correct categories

Figure 2.9 is a unit square representing all of the tokens in §00, broken down horizontally by category match accuracy and vertically by whether the token is in the lexicon. For the standard configuration of StatCCG tested on §00, 42062 of 45422 (92.52%) tokens are correctly labelled. These are represented by the green blocks on the left of the unit square. The remaining 3360 of 45422 (7.48%) tokens have categories that are wrongly assigned. These are represented in red on the right of the

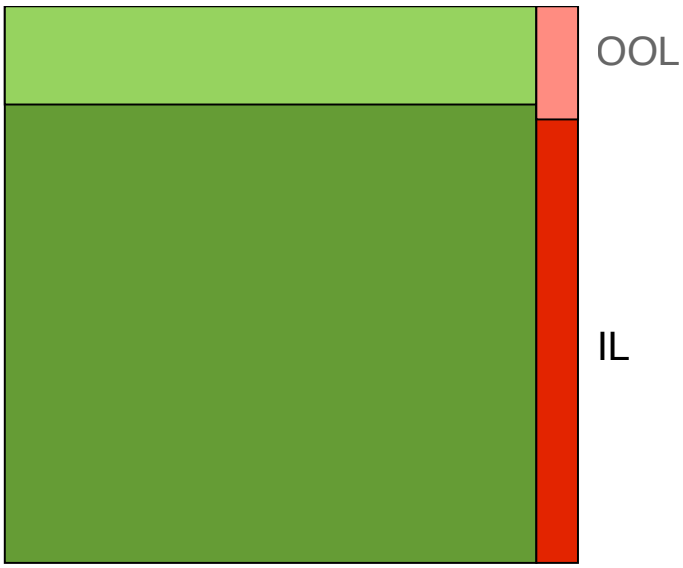


Figure 2.9: Unit square showing composition of category match errors for the StatCCG parser on §00. Green boxes are correct category matches, red boxes incorrect. The darker of each colour represents in-lexicon items (IL) and the lighter out-of-lexicon (OOL).

unit square. While for correct matches, 7336 of 42062 (17.44%) are OOL, for the categories assigned incorrectly, 683 of 3360 (20.33%) are OOL in the baseline lexicon. It is consistent with intuition that OOL items make up a greater proportion of the incorrect matches than the correct ones.

It also means that OOL words are directly responsible for at least 20% of all category match errors in the standard configuration of StatCCG, and indirectly responsible for a further set of errors that derive from those wrong assignments. In some cases, the token could be tagged incorrectly, but still result in the correct dependency structure, either by transforming the erroneous category within the tree using unary rules, or deriving the correct unlabelled dependencies.

This is not to say that these are the only errors that are a direct result of missing entries: a missing lexical entry may also cause the parser to make bad decisions about the surrounding context. Just as well, certain sets of non-matches may be better described as partial or good-enough matches. For example, there is a high degree of redundancy in the prepositional categories of CCGbank, PP/NP and $(NP\backslash NP)/NP$. This is one of the reasons why Hockenmaier and Steedman [2007] calls for evaluation on dependencies, following Carroll et al. [1998] considering it the fairest way to compare CCG

parses against other formalisms and treebanks.[4] It is not entirely accurate to say that these two categories do not match each other, since, with the help of type-changing rules, they end up encoding the same dependency structure. It is a credit to the backoff model of StatCCG that 85% of the pairs in §00 that are absent from the lexicon are nonetheless retrieved accurately. Though the error rate for this best configuration is low, and OOL words account for only 20% of it, the absent lexical entries affect 25% of the sentences, leaving considerable scope for improvement.

2.6 Discussion

This chapter has summarised the related work from word learning to deep lexical acquisition. It has defined the terms in use throughout this thesis, and specified the grammar formalism and parsing tradition in which the word learning task is undertaken. It has also established that that the scope for success is a 20% reduction in error over a very high baseline parsing system. The next two chapters will set out the word learning systems under investigation: first, the two competitive baseline systems, then the chart-based system that is the novel contribution of this thesis.

[4] Of the 52 lower-case words that occur with the prepositional category PP/NP, 50 also occur with $(NP\backslash NP)/NP$ and $((S\backslash NP)\backslash(S\backslash NP))/NP$. The only exceptions are *given*, which occurs as the verbal modifier, and *worth*, which is PP/NP only. This kind of lexical redundancy seems to call for leniency in evaluation.

Chapter 3

Baseline Learning Systems

3.1 Introduction

This chapter fully describes the algorithm and implementation of two baseline word-learning systems. The first learns by brute force, hypothesising all possible categories for an unknown word and allowing the parser to choose the most likely parse, inspired by the work of Watkinson and Manandhar [1999a]. The second uses a small inventory of rules to construct a missing category based on context, an extension to the work of Yao et al. [2009a].

For the purposes of this thesis, we are interested in the circumstances where a sentence of length N is composed of one out-of-lexicon (OOL) word and N-1 in-lexicon (IL) words. This decision is motivated by the phenomena described by Zhang et al. [2005], who set out the parameters of lexicon acquisition and the concomitant problems of evaluation.

The challenges presented by this learning setting are as such: First, it is possible (and quite probable) that the existing lexicon is insufficient to cover exactly the right category types for all of the in-lexicon words. Second, limiting the learning opportunities to only these N-1 cases means only learning from a subset of the sentences in an unlabelled corpus. We must skip over any sentence that contains two unknown words, for example, because it is likely that the category for one depends on the other, so the search space is squared. Zhang et al. [2005] observes the difficulty of adjacent unknown words "potentially allowing almost any constituent to be constructed."

Third, the set of OOL words does not cover the entire set of words we need to learn: we are also searching for additional categories for IL words. It is much harder to determine whether a word/category pair is missing from the lexicon because it is

erroneous, or because it is correct but rare.

We restrict the learning setting to only those sentences of length N for which N-1 of the word tokens are present in the existing lexicon, and only one unknown (OOL) word is present. That word is then treated as the target. Brent effected a similar approach, in that he considered one word at a time within its particular frame. Here, we it would be possible to expand the number of learning opportunities by intelligently splitting the sentential contexts into smaller ones, for example, clauses delimited by semicolons, but that adds another processing step and another source of error. Future work may improve upon the N-1 learning setting by implementing this.

The two word-learning systems described in this chapter operate in this N-1 learning setting, and so provide a comparable baseline with which to compare our novel learning algorithm, which is described in chapter 4.

3.2 Brute Force Word Learning

The first baseline system is the Brute Force (BF) strategy of word learning, which tries all possible categories and allows the parser to choose the highest probability word-category pair by deriving the most likely parse. This system is inspired by the lexical acquisition system of Watkinson and Manandar, which we describe below.

3.2.1 Watkinson and Manandar

One of the early lexical acquisition systems using Categorial Grammar (CG) was that of Watkinson and Manandhar [1999a;b; 2000; 2001a;b]. This system attempted to simultaneously learn a CG lexicon and annotate unlabelled text with parse derivations. Using a stripped-down parser that only utilised the forward- and backward-application rules, they iteratively learned the lexicon from the feedback from online parsing. The system decided which parse was best based on the lexicon, and then decided which additions to the lexicon to make based on principles of compression. After each change, the system reexamined the parses for previous sentences and updated them to reflect the new lexicon.

They report fully convergent results on two toy corpora, but the parsing accuracy of the system trained on natural language data was far below the state of the art. They did present some encouraging findings, however. Primarily they show that Categorial Grammar is a promising setting for artificial language acquisition, because the

structure conflates learning the lexicon and learning the grammar into the same task [Watkinson and Manandhar 1999a]. They also showed that seeding the lexicon with examples of lexical items (closed-class words in their case), rather than just a list of possible categories, increased its chances of converging. It is the focus on annotation that keeps it from being successfully applicable to natural language data.

Watkinson's algorithm has two major flaws. First, the manner of guessing all possible categories, and especially all possible categories for all the words in a sentence, suffers from complexity issues. Second, the algorithm is incapable of learning new category types, which may go unnoticed when the seed is as rich as CCGbank, but severely limits its usefulness as a tool for developing CCG lexicons in previously underserved languages.

3.2.2 Brute Force Algorithm

We implement a generalised version of Watkinson and Manandhar's mechanism for determining the category γ of a single OOL word in a sentence where the rest of the words $C_1 \ldots C_N$ are in the lexicon:

$$\gamma = \arg\max_{\gamma} Parse(C_1 \ldots C_n, \gamma) \tag{3.1}$$

This is equivalent to backing off to the set of all known categories, or replacing the OOL word with the universal POS entry (*); the learner returns the category that maximises the probability of the completed parse tree. We ignore the optimisation and compression steps of the original system, which were shown to increase precision over the course of the learning phase, because they are independent of the contextual word assignment task and could be implemented over any similar sentence-level word learning mechanism. We are interested in comparing the systems over individual sentences.

The BF strategy is almost identical to POS-backoff (see §2.5.1.4), in that the parser is allowed to choose the category that maximises the parse from a list of candidates; essentially it backs off to a single ur-part-of-speech (*). However, if we assume that the target word is unknown, and as such our lexical knowledge is incomplete, it may also follow that our POS knowledge is incomplete. So while this algorithm is more permissive than POS-backoff, it has the freedom to introduce new information to the parser, rather than reinforcing the biases in the original training data. It also allows for instances where POS information is unavailable or incomplete, using it at runtime when necessary, but entertaining new ideas in the training phase.

3.2.3 Worked Examples

In the following worked examples, **X** represents the OOL target word. Any examples of the BF algorithm must take into account a particular parser and lexical model. The model used here is only for illustration purposes. Probabilities have been aggressively rounded to make the examples readable.

(12)

The	**X**	saw	her
$NP/N : 0.084$	$N : 1/\|C\|$	$((S[dcl]\backslash NP)/(S[b]\backslash NP))/NP : 0.500$	$NP[nb]/N : 0.055$
	$NP : 1/\|C\|$	$(S[dcl]\backslash NP)/NP : 0.044$	$NP : 0.024$
	...		$S/(S\backslash NP) : 0.021$

Example (12) shows the problem of learning the word *cat* in a simple declarative context. When presented with this sentence, BF first looks up each token in the lexicon and assigns it a set of possible categories. Each category comes associated with a probability expressing $P(W|C)$. The unknown word **X** is the dummy entry and is assigned every known category with uniform probability ($\forall C : P(\mathbf{X}|C) = 1/|C|$). The parser finds the highest-probability path through these tokens, parsing the sentence as $S[dcl]$. In this case, BF assigns the highest probability to a parse that uses the N entry for **X**, which is the correct category.

(13)

The	cat	**X**	her
$NP/N : 0.084$	$N : 0.01890$	$N : 1/\|C\|$	$NP[nb]/N : 0.055$
	$NP : 0.002$	$NP : 1/\|C\|$	$NP : 0.024$
	$S/(S\backslash NP) : 0.002$	...	$S/(S\backslash NP) : 0.021$

Recovery of a verbal category is more complex, since there are more verbal options available in the set of known categories. In (13) BF finds highest-probability path through *her*:-NP to return the transitive category **X**:-$(S[dcl]\backslash NP)/NP$. More complex verbal categories rank lower because they do not combine as readily with the surrounding nominal constituents to make a complete sentence.

(14)

The	cat	**X**	her	ride
$NP/N : 0.084$	$N : 0.019$	$N : 1/\|C\|$	$NP[nb]/N : 0.055$	$(S[b]\backslash NP)/PP : 0.500$
	$NP : 0.002$	$NP : 1/\|C\|$	$NP : 0.024$	$S[b]\backslash NP : 0.031$
	$S/(S\backslash NP) : 0.002$	...	$S/(S\backslash NP) : 0.021$	$N : 0.006$
				$NP : 0.0005$
				$S/(S\backslash NP) : 0.0005$

The most interesting case is one in which the context is truly ambiguous: in example (14), the words *her* and *ride* could form two valid high-probability interpretations:

either as a single constituent with the category NP, or as two constituents NP and $S[b]\backslash NP$. The learner must rely solely on probability from the parsing model to choose the correct parse, and therefore the category for the target word.

3.2.4 Limitations

The BF word learning method has several shortcomings. Its search space is large, since it entertains all categorial possibilities for the target word, and so its runtime tends to be long. Importantly, it is limited to only those category types it already knows, so in cases where a very limited training corpus is available, it cannot learn all the types in the language. As such, it is subject to the bias of the initial lexicon, and therefore in danger of exaggerating any small errors or imbalances in the model because it cannot create new structures.

3.3 Rule Based Word Learning

The second baseline for word learning is a Rule Based system that applies a defined set of operations over a string of categories to determine the category of the unknown word. It addresses the closed-category-set problem of BF by creating category types on the fly, and is based on the work of Yao et al., described below.

3.3.1 Yao et. al

Yao et al. [2009a;b] developed a learning system based on handwritten translation rules for deducing the category (X) of a single unknown word in a sentence consisting of a sequence of partially parsed constituents (A..N).

Their system was designed to investigate Pullum and Scholz [2007]'s argument against Poverty of the Stimulus, showing that there is much more information in the training set than most researchers see. When the process of learning the lexicon includes typing each entry according to the relations it participates in, larger patterns like auxiliary word order fall out of it naturally. As such, it was based on a small inventory of inference rules that eliminated ambiguity in the ordering of arguments. In addition they limited their learner to CG-compatible parse structures and their constituent strings to length 4. Their argument is that only this minimal bias is needed to learn syntactic structures, including the fronting of polar interrogative auxiliaries and

auxiliary word order (should > have > been), from a training set that did not explicitly contain full evidence for them.

In addition, they imposed a Simple Category Preference (SCP) on the output of the learner: if two or more categories are produced at the same level of complexity, the category with the lowest valency is returned. This bias was appropriate for their level of experimentation, as it eliminated spurious categories, but to be compatible with a full natural-language learning setting, our extension implementation will need a more sophisticated method of ranking output categories.

Another insightful step they used was to generate the possible categories using the full rule set, but then filter them for parses that can be produced entirely using the application rules. This gives the system full generative capability, but limits the size of its lexicon to only the simplest solution. Granted, it will miss those cases where rules besides application are necessary to produce a whole parse, but their artificial data set did not include any of those. It is a clever way of trading off a small amount of coverage for a larger amount of precision. Effectively, they are choosing to only learn from sentences that are fully parseable with application only.

Yao et al. present their findings on a toy grammar only, and outline the challenges to executing the same algorithm on natural language data. They use no statistics, so the system is vulnerable to noise. They label all *N*s and *NP*s using an automatic preprocessing step, which they contend is psychologically plausible, but results in a shallow initial learning curve for natural language settings, since most sentences contain too many OOL words. Their three levels of rules (in figure 3.1) are not sufficient to cover a whole English grammar, but they have shown it to be adequate to capture a considerable subset with good precision. They also restrict their rules so that S/S and $S\backslash S$ are never learned. This will not be much of a help in learning sentential modifiers, but does eliminate the more frequent problem of substrings combining to S prematurely. They are using a partial chart as a learning setting, so some constituents have already been combined, and are opaque with regards to internal structure.

3.3.1.1 Algorithm

Yao et al. [2009b] present a list of handwritten inference rules, reproduced here as figure 3.1. Although their system used the full set of CCG combinators to generate learned categories, it employed a post-processing step to filter spurious categories by checking whether the category participated in a CG-only derivation (using application rules only). This final step goes a long way towards limiting the error introduced by

Level 0 inference rules:

$$B/A \quad \mathbf{X} \quad \rightarrow \quad B \quad \Rightarrow \quad \mathbf{X} = A \quad if\ A \neq S$$

$$\mathbf{X} \quad B\backslash A \quad \rightarrow \quad B \quad \Rightarrow \quad \mathbf{X} = A \quad if\ A \neq S$$

Level 1 inference rules:

$$A \quad \mathbf{X} \quad \rightarrow \quad B \quad \Rightarrow \quad \mathbf{X} = B\backslash A \quad if\ A \neq S$$

$$\mathbf{X} \quad A \quad \rightarrow \quad B \quad \Rightarrow \quad \mathbf{X} = B/A \quad if\ A \neq S$$

Level 2 inference rules:

$$X \quad A \quad B \quad \rightarrow \quad C \quad \Rightarrow \quad \mathbf{X} = (C/B)/A$$

$$A \quad B \quad X \quad \rightarrow \quad C \quad \Rightarrow \quad \mathbf{X} = (C\backslash A)\backslash B$$

$$A \quad X \quad B \quad \rightarrow \quad C \quad \Rightarrow \quad \mathbf{X} = (C\backslash A)/B$$

Level 3 inference rules:

$$\mathbf{X} \quad A \quad B \quad C \quad \rightarrow \quad D \quad \Rightarrow \quad \mathbf{X} = ((D/C)/B)/A$$

$$A \quad B \quad C \quad \mathbf{X} \quad \rightarrow \quad D \quad \Rightarrow \quad \mathbf{X} = ((D\backslash A)\backslash B)\backslash C$$

$$A \quad \mathbf{X} \quad B \quad C \quad \rightarrow \quad D \quad \Rightarrow \quad \mathbf{X} = ((D\backslash A)/C)/B$$

$$A \quad B \quad \mathbf{X} \quad C \quad \rightarrow \quad D \quad \Rightarrow \quad \mathbf{X} = ((D\backslash A)\backslash B)/C$$

Figure 3.1: Inference rules from Yao et al. [2009b].

the inference process, but ensures that the process will not scale up to handle the full range of category types, some of which are only used with composition rules.

For example, one of the Level 3 inference rules specifies the order of the arguments in the deduced category:

$A\ \mathbf{X}\ B\ C \rightarrow D \Rightarrow \mathbf{X} = ((D\backslash A)/C)/B$

Without this inductive bias the learner would have to deal with the ambiguity of the options $((D/C)/B)\backslash A$ and $((D/C)\backslash A)/B$ at minimum.

3.3.1.2 Shortcomings

The central problem with Yao et al.'s algorithm, which they acknowledge, is their right-combining rule preference. As stated in figure 3.1, the second- and third-level rules have only one result, which is the one that derives from recursively applying only the level 1 rules, rather than both the level 0 and 1 rules as stated in the paper. This shortcoming is conflated with the right-combining rule, as illustrated by the figures 3.2, 3.3, and 3.4 below.

Under the established inference rules, 'X 's doll slept' is of type (1) below, and therefore subject to rule 3.1 If we expand the view to allow complex categories of the kind acknowledged in the level 0 rules, we can see that this sentence is more accurately

Peter	slept	with	Mary			
NP	$S\backslash NP$	X	NP	$\rightarrow$	S	
	$S\backslash NP$	X	NP	$\rightarrow$	$S\backslash NP$	1.1
		X	NP	$\rightarrow$	$(S\backslash NP)\backslash(S\backslash NP)$	1.1
		X		$\rightarrow$	$((S\backslash NP)\backslash(S\backslash NP))/NP$	1.2

Figure 3.2: 'Peter slept X Mary' is derivable.

Sally	's	doll	slept			
X	$(NP/N)\backslash NP$	N	$S\backslash NP$	$\rightarrow$	S	
X	$(NP/N)\backslash NP$	N		$\rightarrow$	NP	0.2*
X	$(NP/N)\backslash NP$			$\rightarrow$	NP/N	1.2
X				$\rightarrow$	NP	0.2*

Figure 3.3: 'X 's doll slept' is not derivable because rule 0.2 is left-combining.

of form (2). And in such a case, $X = C$.

X	A	B	C	$\rightarrow$	D	(1)
X	$(A/B)\backslash C$	B	$D\backslash A$	$\rightarrow$	D	(2)

Yao et al. have enumerated a small number of elegant rules for learning unknown words in a restricted language, but by restricting the derivations to only the right-combining ones, they have made a whole set of derivations impossible. This is not a problem for their experimental space, but to deal with natural language on the scale with which we are concerned, we must implement the algorithm as described by their diagram only. The way to do this is to only specify the level 0 and 1 rules, and recursively apply these rules to larger contexts, not limiting the constituent sequence by length, and not enumerating those as higher-level rules. This modification was obvious to the original experimenters [Yao et al. 2009b], and in a footnote, they plan to relax the right-combining restriction and allow S/S and $S\backslash S$ as future work.

The main problem with Yao's original algorithm is that it does not scale to the full complexity of parsing options afforded by a natural-language model. Once the complexity restriction and CG output-filters are removed, the Simple Category Preference is no longer a reasonable assumption. For almost any sentence we can find an arrangement of constituents that allow for an NP interpretation of the missing word. Therefore in order to make this approach scalable, we must implement a statistical model to calculate the relative probabilities of each of the interpretations.

Sally	's	doll	slept			
X	$(NP/N)\backslash NP$	N	$S\backslash NP$	$\rightarrow$	S	
X	$(NP/N)\backslash NP$	N		$\rightarrow$	$S/(S\backslash NP)$	1.2
X	$(NP/N)\backslash NP$			$\rightarrow$	$(S/(S\backslash NP))/N$	1.2
X				$\rightarrow$	$((S/(S\backslash NP))/N)/NP$	1.2

Figure 3.4: Right-combining derivation of 'X 's doll slept'.

The right-combining preference, which in small language-acquisition applications is seen as a useful inductive bias, does not scale well to practical natural language applications. Rather, it limits the productivity of an otherwise competent learning method. In addition, the recursive rules only encode the functionality of the application rules. A full model would include composition at the least. There are also the unary transformations and non-combinator rules as found in CCGbank.

Rather than implement all of these as extensions on Yao's rules (since much of the elegance of the original has now been lost) we propose a separate algorithm that addresses each of these challenges in a complete solution.

3.3.2 Rule Based Algorithm

The RB learner operates similarly to the BF learner, in that it starts with a lookup of all known words in the sentence. It begins by treating the shortest constituent sequence available and moves onto longer sequences upon failure.

We extend the work of Yao et al. [2009a;b] to create a generalised word-learner that is capable of deducing the CCG category of an unknown word from context. Our contributions are: 1) adding full recursion to the algorithm, 2) removing the complexity restriction on learning contexts, 3) implementing the learner on top of a full statistical CCG parser framework, 4) removing the CG filter on completed parses and 5) modifying the return-simplest-category heuristic to a probability-ranked list of candidate categories. These improvements have been partially anticipated by the original authors:

> Using statistical patterns in the input language, it may also be possible to relax some of the assumptions presented here. [The system's limitations] mark one direction in future work: developing the algorithm further to make use of statistical learning methods, and evaluating it on real data. [Yao et al. 2009a]

$\textsc{derive}([C_1...C_n], \beta, \gamma)$

if $\beta = \emptyset$

then return (γ)

else if $C_1 = C_n = X$

then $\begin{cases} \gamma = \gamma + \beta; \\ \textsc{derive}(X, \emptyset, \gamma) \end{cases}$

else $\begin{cases} \textsc{derive}([C_2...C_n], \beta \backslash C_1, \gamma) & \textbf{if } C_1 \notin S, X \\ \textsc{derive}([C_1...C_{n-1}], \beta / C_n, \gamma) & \textbf{if } C_n \notin S, X \\ \textsc{derive}([C_2...C_n], A, \gamma) & \textbf{if } \beta \equiv B \textbf{ and } C_1 \equiv B/A \textbf{ and } C_1 \notin S, X \\ \textsc{derive}([C_1...C_{n-1}], A, \gamma) & \textbf{if } \beta \equiv B \textbf{ and } C_n \equiv B \backslash A \textbf{ and } C_n \notin S, X \end{cases}$

Figure 3.5: RB algorithm generalised from Yao et al. [2009a], where $[C_1...C_n]$ is a sequence of categories, one of which is X, β is a result category, and γ is the (initially empty) target category set.

Their rules were effective for their learning setting, but for the purposes of this thesis we have implemented a generalised version of the recursive algorithm for use in wide-coverage parsing. This algorithm is outlined in figure 3.5. It takes a sequence of categorial constituents, all known except one (**X**), and builds a candidate set of categories (γ) for the unknown word by recursively applying Yao's Level 0 and Level 1 inference rules.

To accommodate all the parse options open to a natural-language-scale lexicon, the recursive algorithm is run over the full set of partial parses, beginning with the shortest sequence. This algorithm begins by covering the whole sentence with sequence $[C_1...C_n]$. Each level of recursion acts on a sequence shorter by exactly one constituent. It exits when there is only one constituent left, which must be the target word. Where the original learner uses a heuristic to select the best solution, the generalised RB system returns the set of category of the lowest valency. This setting could be relaxed to create a system that is more competitive in terms of precision, but much slower. The various methods of using and evaluating this answer set are discussed in subsequent chapters.

It is important to note that the rule-based method has high coverage because it always produces a category for any situation you give it, except in the case where a sequence of length ≤ 4 cannot be found. This preference for recall over precision is at

	P	R	F	T(m)
$N\|NP$	33.68	33.68	33.68	-
Original	16.42	16.42	16.42	14
Generalised	39.01	37.20	38.08	15

Table 3.1: Precision and recall over category matches in MG1, comparing Yao's original algorithm to our generalised RB algorithm against a baseline of guessing N or NP. The final two columns report Correct/Attempted.

odds with the goal of this thesis, and as such, it represents a low baseline.

3.3.3 Evaluating the Generalised RB Algorithm

To compare Yao's original algorithm to our generalised version, we run both over the words in volume 1 of the McGuffey corpus (MG1), which is explained in full in chapter 6. A full third of the lexical entries in the corpus are N or NP, and we use this as our comparative baseline for this experiment. Table 3.1 lists precision, recall and F-score figures results for the original and generalised RB systems, as well as the time they take to process the 546 sentences in the corpus. It shows that the generalised algorithm achieves double the F-score of the original, while retaining its speed.

Table 3.2 breaks down the results by the category type of the correct word labels, for the most common types. First, it shows that in practice our generalised RB system increases the number of correct labels assigned correctly, while decreasing the number of labels attempted. This translates to the gain in F-score shown in table 3.1. In addition, the generalised learner also shows improvement in several categories for which the original reported zeroes, which tend to be more complex categories that would have been filtered out by the simple-category preference, or blocked entirely by the the right-combining preference. This shows that our efforts to generalise the word learner of Yao et al. have produced a word-learning system that has enough expressive power to beat an naive baseline, and so serves as a reasonable comparator for our novel word learning system introduced in the next chapter.

Even though this system is being used as a baseline comparator for the Chart Inference word-learning system, the rule-based approach is not itself a dead end. Further work could extend the RB algorithm to address most of its remaining shortcomings, with the potential for an expressive and reliable CCG word-learner that is faster than chart-based learners.

Frequency	C	Original	Generalised
828	NP	102/225	181/211
708	N	78/196	137/191
578	$NP[nb]/N$	0/170	82/165
397	PP/NP	2/107	68/103
314	N/N	8/85	38/82
113	$(S[dcl]\backslash NP)/(S[b]\backslash NP)$	0/28	18/28
107	$(S\backslash NP)\backslash(S\backslash NP)$	0/13	0/11
105	$(S[dcl]\backslash NP)/NP$	0/26	20/25
90	$(S[b]\backslash NP)/NP$	0/30	20/28
72	$(S\backslash S)/NP$	0/1	1/3
71	$(S[to]\backslash NP)/(S[b]\backslash NP)$	0/15	9/15

Table 3.2: Category accuracy comparison between Yao's original algorithm and our generalised version, for the top 11 most frequent category types in MG1.

3.4 Comparing the Baseline Learning Systems

We compare the BF algorithm to our generalised RB algorithm, with respect to their ability to learn a 40-item toy lexicon, which is explained in full in chapter 5. The only functional difference between the two algorithms in this learning setting, since they are both statistical and using the same model, is in their ability to generate new category types. Both learners begin with a seed lexicon that consists of only the *NP*s entries of the target lexicon. This setting is motivated jointly by Yao et al. [2009b] and Watkinson and Manandhar [1999a]: they both cite NPs as the low-hanging fruit of lexicon acquisition, making them the logical starting point for a seed lexicon. The test sentences are randomly generated from a simple PCFG over the lexicon, and are always presented to the learners in the same order. For this experiment we sort the corpus by sentence length.

Figure 3.6 shows the learning curves of the two methods as they traverse the training sentences, in terms of similarity of the learned lexicon to the target lexicon [1]. The BF method performs poorly, as it is incapable of learning category types absent from the seed lexicon and therefore only acquires erroneous *NP* entries. This is an inherent property of the algorithm, since the search space only includes known types. As such,

[1] See equation 5.1.

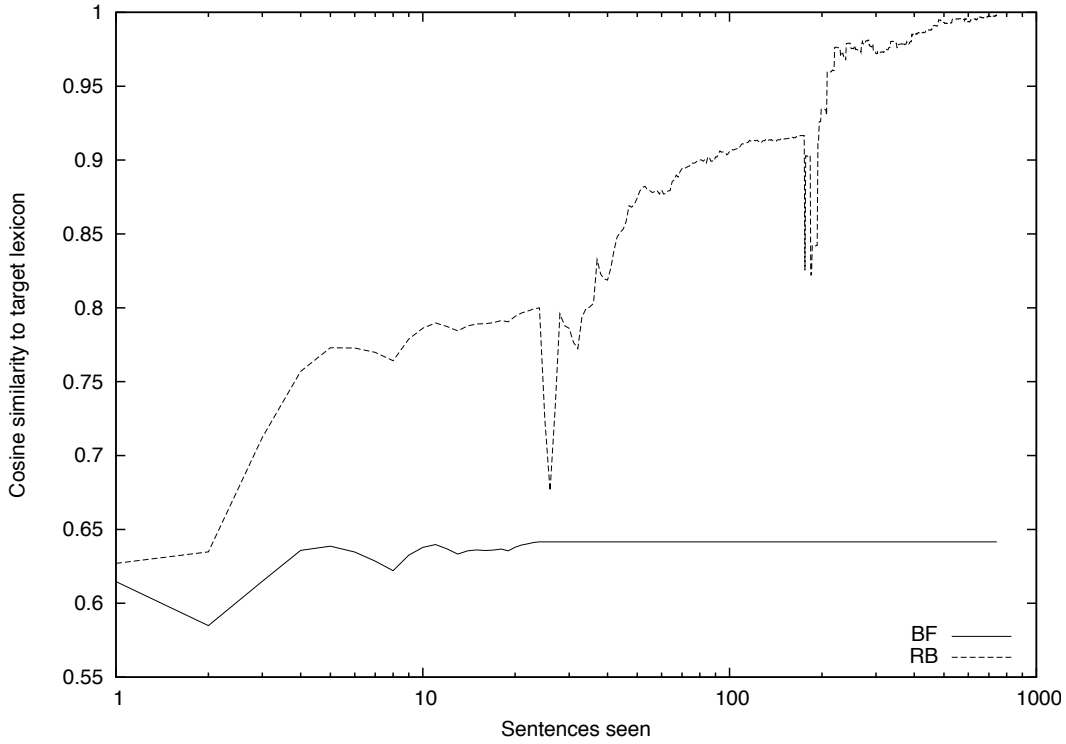


Figure 3.6: Learning curves comparing the generalised rule-based learner (RB) to Brute Force (BF), using a seed lexicon of only NP entries.

NP is the only option available, so the maximum achievable cosine is 0.64, representing six correct lexical items out of 40.

The rule-based learner achieves a cosine similarity to the target lexicon of 0.99 after only 458 sentences. The two major dips in the graph correspond to the first sighting of longer sentences in the training corpus. Sentences 1-25 are of length 2, and as such contain only *NP*s and $S\backslash NP$s. The first of the three-word sentences, 26 and 27, are the first introduction of transitive verbs and determiners, respectively. In both cases the learner deduces the correct category, but since it only has one word of that type so far, the lexical probability $P(W|C) = 1$.

The second dip is caused by the beginning of the four-word sentences. It is the first time the learner encounters adjectives and ditransitive verbs, and it also assigns them the correct categories. After it has seen enough instances of different words, the similarity recovers, and then increases smoothly to convergence.

This highlights a bias in using cosine similarity to evaluate lexicon quality: the learner is better off having no entries for a category than just one. This does not necessarily reflect the lexicon quality when it comes to applying it to a parsing task. As such, we only use the cosine-similarity metric to evaluate the convergence of toy

lexicons to a known target. Real language data, apart from not having a gold-standard lexicon for evaluation, needs a more complex measure of lexicon quality that takes into account the effectiveness when used to parse real sentences.

This comparison of the two baseline methods shows that the ability to invent new category types is integral to success in cases where the seed lexicon does not contain at least one instance of all categories needed to parse the corpus. As this is the case in many of the learning tasks we wish to attempt, the ideal learner needs this ability to outperform the BF baseline, and needs a sophisticated method for ranking possible solutions, including intermediate ones, to outperform the RB baseline.

3.5 Discussion

Four baseline methods are used for comparison in this thesis. The simplest of these is the $N|NP$ baseline, where all words are assigned $N|NP$ and either category is marked as correct. The second is Part-of-Speech backoff, where the target word is assigned the category type that is most likely for that part of speech. This metric can be affected by whether the POS tags have been assigned automatically or hand-labelled. The next baseline method in order of complexity is Brute Force, based on Watkinson and Manandhar, allowing the parser to choose whichever category type maximises the probability of the whole parse. Finally, we compare our system to the Rule Based learner, a probabilistic extension of Yao et al., which applies an ordered set of rules to determine the category type of the target word. It is important to note that the two baselines learning systems BF and RB are not restricted to using the correct parse, and as such are subject to parse errors and vagaries of the lexicon and model, meaning that the highest-probability category for the target does not necessarily participate in the correct parse, since the correct parse is treated as unknown.

Extant erroneous lexical entries also cause problems for learning new lexical entries, because they tend to interfere in the parsing process due to their relative probabilities being higher than those of more common categories. It is vital to note that $\sum_C P(W|C) \neq 1$. The type of generative model we are concerned with operates over lexical probabilities of the word given the category, which means that the total probability mass for each category is shared by all words of that category. Whether a word has more than one category is not reflected in the lexical probabilities. Any process that has to choose a category out of several candidates will prefer the one with the most absolute probability mass, not the one that is most common. This often results in

erroneous categories participating in ambiguous parses.

These word-learning methods will be used as baseline against which to compare the performance of our more sophisticated learner, described in the next chapter. We seek to improve on those weaknesses which we have identified in each method, but some problems are integral to the domain of word-learning and will challenge all learners irrespective of their form.

Chapter 4

Chart Inference

4.1 Introduction

The major contribution of this thesis is the word-learning algorithm we call Chart Inference (CI). It employs a two-stage chart parsing process to derive the category of an unknown word based on the surrounding constituents. This chapter explains the algorithm in full, as well as details of its implementation with the StatCCG parser.

4.2 Motivation

In our quest to answer Brent's challenge in the context of wide-coverage parsing, we need an algorithm that is able to handle sources of error and scale to contemporary parsing tasks. Both Watkinson's and Yao's experiments were fully convergent over a toy dataset, but did not scale to realistic corpora. Watkinson attempted to learn from the LLL corpus [Kazakov et al. 1998], but attributed the failure to the small amount of training data relative to the test corpus, and the naive initial category set. Yao's method was only ever designed as a proof-of-concept to show how much of the language can be learned from partial evidence, and was not meant to be run in earnest in a real-world learning setting. For one, Yao's rules do not cover the full set of partial parse conditions. Moreover, they do not allow for partial parses to be reanalysed within the learning framework.

To that end, we have developed a learning algorithm that is capable of operating within the N-1 learning setting adopted by the two baseline systems, that is able to invent new category types, and that is able to take advantage of the full generality of CCG. This chapter shows that CI performs as well as the previous two systems on a toy

corpus, and the subsequent chapters explore how it scales to natural language domains.

Both Watkinson's and Yao's systems are targeted to incremental learning, treating one sentence at a time and updating the lexicon before starting on the next. They ignore sentences that are not immediately informative and come back to them later. However, in order for the system to perform competitively in natural language applications, it has to perform well on every sentence it parses, returning either a high-quality parse or no parse. This translates to high precision, at the expense of recall. A method's precision will be a good predictor of its performance on real language data, and its efficiency will be a good predictor of scalability.

4.3 Related Work

The work most closely related to Chart Inference, seen as a two-stage parsing process employing both bottom-up and top-down steps over a partial chart, is the unglamorous task of error repair. The field was discussed in brief in Chapter 2, but two older systems are particularly relevant to our work.

Mellish [1989] established a two-stage bidirectional chart parser for diagnosing errors in input text. His system relied heavily on heuristic rules, and the only evaluation he did was on number of cycles needed for each type of error, and number of solutions produced. It was designed for use in producing parses where the original parser failed, dealing with omissions, insertions, and misspelled/unknown words. The only method used to rank the possible solutions was heuristic scores.

Kato [1994] implemented a revised system that used a generalised top-down parser, rather than a chart, and was able to get the number of cycles to decrease. In both cases the evaluation was only on a toy corpus, and neither Mellish nor Kato evaluated on whether the systems diagnosed the errors correctly, or whether the solution they offered was accurate. They also had to deal with cases where the error was ambiguous, for example, where an inserted word could be interpreted as a misspelling. Min and Wilson [1998a;b] went further, implementing an integrated bidirectional chart parsing system with "lexical, syntactic, surface case, and semantic processing."

Our statistical implementation of CI could be considered a modern extension of Mellish's work. Where he uses the two-stage parsing process to complete malformed parses, we use it to diagnose unknown lexical items. In addition, we scale the process to a full grammar and parser, and to modern lexical corpora which he left as future work.

4.4 Algorithm

The CI algorithm is best contextualised in the tradition of chart parsing. The formalism-independent CKY algorithm [Kasami 1965; Younger 1967; Cocke 1969] is expressed in figure 4.1; its central principle is that when two sisters are known, their mother can be produced by consulting the rules of the grammar, building a parse from the bottom up [1]. The formulation we give here is non-standard, but designed to conform to a human-readable chart and make intuitive sense in both directions, with the terminal cells at $[i][i]$.

```
for i = 1 to n:
  chart[i][i] = set of terminals expanding to w_i
for k = 2 to n:
  for j = k-1 to 1: -- decrementing
    for i = j to k-1:
      for all A -> B C in G,
        such that there is a B in chart[i][j] and a C in chart[k][i+1],
        if there is no A in chart[k][j], insert A into chart[k][j]
```

Figure 4.1: Bottom-up CKY algorithm with generative model G, adapted from Hockenmaier [2003], where *i* is the start position, *j* is the end position and *k* is the split.

```
chart[n][0] = set of preterminals (root)
for k = n to 2: -- decrementing
  for j = 1 to k-1:
    for i = k-1 to j: -- decrementing
      for all A -> B C in G,
        such that there is an A in chart[k][j],
        if there is no B in chart[i][j], insert B into chart[i][j]
        if there is no C in chart[k][i+1], insert C into chart[k][i+1]
```

Figure 4.2: Top-down chart parsing algorithm with generative model G, where *i* is the start position, *j* is the end position and *k* is the split.

[1] All of the algorithms given here are expressed in terms of a binary grammar, but can be generalised to a non-binary grammar formalism.

```
chart[n][0] = set of preterminals (root)
for a = 1 to n:
  for b = 1 to i:
    chart[a][b] may be either a set of (pre)terminals or the empty set
for k = n to 2: -- decrementing
  for j = 1 to k-1:
    for i = k-1 to j: -- decrementing
      for all A -> B C in G,
        such that there is an A in chart[k][j] and a B in chart[i][j]:
        if there is no C in chart[k][i+1], insert C into chart[k][i+1]
      for all A -> B C in G,
        such that there is an A in chart[k][j] and a C in chart[k][i+1]:
        if there is no B in chart[i][j], insert B into chart[i][j]
```

Figure 4.3: Top-down Chart Inference algorithm with generative model G, where *j* is the start position, *k* is the end position and *i* is the split.

The top-down chart parsing process is in a sense the opposite of CKY. Given the result, both daughters are produced using the model. It is shown in figure 4.2.

Chart Inference combines elements of both bottom-up and top-down chart parsing. First, a partial parse is produced from the bottom up, as far as possible given that one terminal is unknown. This is expressed in the second through fourth lines of figure 4.3. The second half operates from the top down, filling in daughters where possible, and expressed starting at the fifth line of figure 4.3.

The learning step begins by presenting the parser with any sentence that contains all in-lexicon words but one. The parser produces a partial bottom-up parse chart using the statistical parse model, which contains a seed lexicon, a set of CCG combinators, and an additional set of unary and binary rules learned from the training corpus. The learner takes this partial chart and fills the top right cell with a distribution for the result category based on the end punctuation, which is thereafter ignored[2].

Using this partial chart that contains at least one entry for every terminal cell (except the one OOL target cell) and at least one entry for the result, the learner steps through the partially populated chart from the top. For the top-down process, the stan-

[2]For simple corpora, only *S* is required, but natural language corpora necessitate a distribution over all result types, including noun phrases and fragments.

CCG Combinator					**Inverse Combinator**							
A/B	B	$\rightarrow$	A	$(>)$	1	**X**	B	$\rightarrow$	A	$\Rightarrow$	A/B	if v(B) ≤ 1
					2	A/B	**X**	$\rightarrow$	A	$\Rightarrow$	B	
B	$A\backslash B$	$\rightarrow$	A	$(<)$	3	**X**	$A\backslash B$	$\rightarrow$	A	$\Rightarrow$	B	
					4	B	**X**	$\rightarrow$	A	$\Rightarrow$	$A\backslash B$	if v(B) ≤ 1
A/C	C/B	$\rightarrow$	A/B	$(>B)$	5	**X**	C/B	$\rightarrow$	A/B	$\Rightarrow$	A/C	
					6	A/C	**X**	$\rightarrow$	A/B	$\Rightarrow$	C/B	
$C\backslash B$	$A\backslash C$	$\rightarrow$	$A\backslash B$	$(<B)$	7	**X**	$A\backslash C$	$\rightarrow$	$A\backslash B$	$\Rightarrow$	$C\backslash B$	
					8	$C\backslash B$	**X**	$\rightarrow$	$A\backslash B$	$\Rightarrow$	$A\backslash C$	

Figure 4.4: Derivation of inverse combinators.

dard combinators have to be reformulated to take an argument and a result as inputs, rather than two arguments as in the standard bottom-up case. In addition, the learner has access to the non-combinator rules from the parse model, which have been similarly inverted for top-down use. The inverse combinators and model rules are discussed in the next section. This process continues until the target cell has been filled, and the ranked set of categories is returned. We provide worked examples below in 4.5.

4.4.1 Inverse Combinators

In order for CI to access the generative model from both directions, we must perform a small amount of reinterpretation of the grammar and model. The bottom-up model is drawn intact from the underlying parser. The top-down model calculations are straightforward; we simply use the same rules. However, it is useful to translate the CCG combinators into their top-down formulations, which give the solution for the missing daughter **X** when the other sister and the result are specified.

Figure 4.4 sets out the inventory of inverse combinators used in the top-down learning step. Each standard binary CCG combinator motivates two inverse combinators: one for each possible missing item, denoted by **X**. In the two permissive instances where the sister category's form is the unrestricted B, we limit the sister's valency (v) to 1, in order to keep the learner from generating spurious categories that could result from these two rules being over-applied.

These inverse combinators lead to some ambiguity, as any category of the form A/B is also of the form A. As such, any time rule 5 is used, rule 1 could also be used, so we order the rules such that inverse composition is attempted first, and then inverse

application as a last resort. In addition, the probabilities calculated with respect to the model should sift bad rules to the bottom.

In addition to the combinators, the algorithm has access to all the binary and unary rules in the model, read off the model in the inverse direction. This means that type-raising rules and useful unary rules can be used to generate an inference, but they are not generalised, as that would result in overgeneration. As such, only those unary rules that have been instantiated can be accessed by the learner. To prevent overgeneration, a condition is placed on two of the inverse application rules (1 and 4 in figure 4.4). In those cases, the known daughter must be a simple category type (with valency less than or equal to 1) in order for the inference to go ahead. Without this restriction, these rules are too permissive, allowing any result/daughter combination to yield a possibly unjustified category.

The full extent of the category model at present is a restriction against returning the punctuation category (.) as the highest ranking category at the terminal node. The learner is currently (and may always be) incapable of identifying missing *conj* categories. This is because it is only capable of doing unary and binary relations, and coordination is treated as ternary by the StatCCG and StatOpenCCG parsers. *neg* is only possible because the model rules are used along with the combinators. A more developed theory of category well-formedness would benefit the generation algorithm, either as a filter on the end of the production process, or as a set of guidelines that influence the generation online.

4.4.2 Statistical Model

The probability that the target has a given category is calculated as the greater of the right- or left-headed derivations, according to figure 4.5. This is derived directly from the head-driven statistics of StatCCG discussed in chapter 2, and the figures are taken directly from the trained StatOpenCCG model without adjustment. At training time, the StatOpenCCG parser creates a head-dependency model from the training corpus, in which we can look up the values for the expansion probabilities.

Unary rules are calculated as $P(result) * P(exp = un|result) * P(C|exp = un, result)$, which is the product of the probability of the result category, the probability of that result having a unary expansion, and the probability of the category C expanding from the result. This operation is executed on every cell upon the first visit. Where a value is unavailable, it backs off to a pre-specified value (default 0.0001). The system requires

$$P(target = C|R,S) = \max \left\{ \begin{array}{r} P(HeadRight|R) \\ P(HeadLeft|R) \end{array} \right\}$$

$$P(HeadRight|R) = \left\{ \begin{array}{r} P[outside](R)* \\ P[inside](S)* \\ P(exp = left|R)* \\ P(C|R,exp = left)* \\ P(S|R,exp = left,C) \end{array} \right\}$$

$$P(HeadLeft|R) = \left\{ \begin{array}{r} P[outside](R)* \\ P[inside](S)* \\ P(exp = right|R)* \\ P(S|R,exp = right)* \\ P(C|R,exp = right,S) \end{array} \right\}$$

Figure 4.5: Charted head-dependency probability calculation of a category C, derived from one Result category (R) and one Sister category (S).

a pruning parameter that limits each cell to the top N most probable categories. For most of the experiments in this thesis, we set the pruning threshold to 10, to limit the search space and complexity[3].

4.5 Worked Examples

In this section we work through all the steps of CI on sample sentences to show its strengths and weaknesses. Three charts are presented in the subsequent pages. They represent formalised grammars, with only the interesting lexical items included, for reasons of brevity. Full parsers would result in much fuller charts, where here we have only included as many lexical entries in a cell as are interesting for our purposes, and represented the remainder as ellipses. In each chart, the **X** represents the unknown word. The first action is the filling of the top-right cell [N,1]. In this case we have abridged the end punctuation model and assumed $P(S[dcl]|.) = 1$.

Next, the terminal cells are filled by looking up the words in the lexical model.

[3]Separate testing on natural language corpora has shown the average rank of correct tags in the category set to be 1.4.

Then all possible non-terminals are filled using the bottom-up parser, which calculates the probabilities as well. These cells are represented with grey shading, signifying that they are impermeable to the top-down parser. These entries are then only used as inputs to the reverse combinators. An extension to this process would allow new entries to be added to the shaded cells, in order to recover partial parses that were unavailable to the parser[4]. This, however, would mean entertaining new categories for known words as well as longer spans, which should be undertaken with caution and a good sense of the quality of the existing parsing model.

Table 4.1 illustrates how CI works through the sentence *The cat saw her* . We assume for this example that *cat* is the OOL word, and that the correct category is N. For clarity and space, only a small selection of the possible categories are displayed in the chart. Three terminal cells are filled by the bottom-up parsing process; they are shaded grey. Two of the known words are lexically ambiguous: *saw* has entries in the lexicon as both a transitive and a control verb. *her* includes the determiner and noun phrase senses, as well as the type-raised subject category $S/(S\backslash NP)$. Cell (4,3) covering *saw her* is filled by the bottom-up parser as well, entertaining four possible categorial types.

The top-down phase begins by filling the root cell (4,1). Here we show only the canonical declarative form $S[dcl]$, as this example has been simplified for illustrative purposes. The CI algorithm begins with $k = 4$, $j = 1$, $k = 3$, using the result cell (4,1) and the terminal cell (4,4) to fill the non-terminal (3,1). Two entries are produced for that cell: $S[dcl]/NP$ is generated from using inverse combinator (1) on the noun phrase sense of *her*: $\mathbf{X}\ \ NP \rightarrow S[dcl] \Rightarrow S[dcl]/NP$, and $S[dcl]\backslash NP$ results from the inverted model rule $\mathbf{X}\ \ S/(S\backslash NP) \rightarrow S[dcl] \Rightarrow S[dcl]\backslash NP$.

Next, when the split (i) is decremented, the same result cell (4,1) is treated with (4,3) to fill (2,1). A number of entries are generated, the most probable of which are shown. NP is generated from the most probable entry in (4,3) using inverse combinator 3: $\mathbf{X}S[dcl]\backslash NP \rightarrow S[dcl] \Rightarrow NP$. $NP[nb]$ is from a model rule, and N is a product of a unary rule on NP. The third step deals with the two sister cells (1,1) and (4,2), this time with the right sister as the gap. Inverse combinator 4 is the relevant one, yielding $NP/N\ \ \mathbf{X} \rightarrow S[dcl] \Rightarrow S[dcl]\backslash(NP/N)$. Having exhausted i, j is now incremented and the whole process begins again with (4,2) as the result cell.

The remainder of the white cells are derived from the top down, until cell (2,2) is filled. The solution is presented as a ranked list of possible categories for the target

[4]Some investigation was made in this direction, but parse times became prohibitive.

Table 4.1: Chart illustrating CI recovering a noun: Grey boxes are filled bottom-up by the parser; white boxes top-down by the learner.

1	2	3	4	
NP/N : 0.08428	NP : 2.01E-5 N : 1.85E-6 $NP[nb]$: 6.97E-8 ...	$S[dcl]\backslash NP$: 2.90E-5 $S[dcl]/NP$: 2.90E-10	$S[dcl]$: 1.0	1
The	N : 6.50E-9 $(S\backslash NP)/(S\backslash NP)$: 9.61E-11 $(S[dcl]\backslash NP)/(S[dcl]\backslash NP)$: 1.20E-11 ...	$(S[dcl]\backslash NP)/NP$: 1.35E-7	$S[dcl]\backslash(NP/N)$: 1.00E-4	2
	X	$((S[dcl]\backslash NP)/(S[b]\backslash NP))/NP$: 0.50000 $(S[dcl]\backslash NP)/NP$: 0.04348	$S[dcl]\backslash NP$: 5.85E-5 $(S[dcl]\backslash NP)/(S[b]\backslash NP)$: 8.65E-8 $((S[dcl]\backslash NP)/(S[b]\backslash NP))/N$: 2.73E-11 $(S[dcl]\backslash NP)/N$: 2.38E-12	3
		saw	$NP[nb]/N$: 0.05467 NP : 0.02439 $S/(S\backslash NP)$: 0.02124	4
			her	.

word, the most probable of which, N, is correct. The statistical capability of CI makes it robust to ambiguity. In this example the highest-ranking lexical category for *her* is $NP[nb]/N$, but the next highest category NP is preferred in the derivation of the highest-ranking category for the unknown word **X**.

Table 4.2 works through the same sentence, but now the target word is *saw*, which has the transitive category $(S[dcl]\backslash NP)/NP$. In this case the Noun/Determiner ambiguity is also present, and lacks corroborating evidence from the verb, since it is unknown. So the erroneous category $(S[dcl]\backslash NP)/NP[nb]/N$ is possible, but CI produces the correct category as the highest ranked one, due to the influence of the non-lexical portion of the model that favours verbal categories with nominals, rather than determiners, as arguments.

Table 4.3 deals with the more complex sentence *The cat saw her ride* . In this case, the categories for *her*, *ride*, and *her ride*, corresponding to cells (4,4), (5,4), and (5,5) are truly ambiguous. As such, the learner relies entirely on the judgements of the model, which can be swayed by the nature of the training corpus. In this case, the highest-probability interpretation yields the target category as the plain transitive $(S[dcl]\backslash NP)/NP$, rather than the control verb $((S[dcl]\backslash NP)/(S[b]\backslash NP))/NP$, which manifests as third in the ranked list, although the NP sense of *her ride* does not rank in the top four categories in (5,4).

4.6 Discussion

The Chart Inference algorithm is the novel contribution of this thesis: it creates category types by applying inverted CCG combinators and model transformations top-down on a partial chart.

There are several major differences between our learner and the baseline methods. First, we are using a full parser, complete with all the combinatory rules. Second, we make no attempt to constrain the lexicon according to any principles of compression. We are depending on the model to give us fairly dependable lexical entries, and for the erroneous ones to have probabilities low enough to keep them out of the top-ranking parses. In addition, we are not keeping track of parses or refiguring them based on subsequently learned lexical entries.

BF has been shown to be effective over a toy corpus, but does not scale as well to natural language corpora. It has access to all the category types yet seen in the language, but is also stressed with a large search space. The RB system was not designed

Table 4.2: Chart illustrating CI recovering a verb: Grey boxes are filled bottom-up by the parser; white boxes top-down by the learner.

1	2	3	4	
$NP[nb]/N$: 0.08428	$NP[nb]$: 0.00138	$S[dcl]\backslash NP$: 2.90E-5 $S[dcl]/NP$: 2.90E-10	$S[dcl]$: 1.0	1
The	N : 0.01890 NP : 0.00174 $S/(S\backslash NP)$: 0.00152	$(S[dcl]\backslash NP)/NP$: 1.35E-7	$S[dcl]\backslash(NP/N)$: 1.00E-4	2
	cat	$(S[dcl]\backslash NP)/NP$: 2.21E-9 $(S[dcl]\backslash NP)\backslash NP[nb]$: 6.06E-19 $(S[dcl]/NP)\backslash NP[nb]$: 4.00E-23 ...	$S[dcl]\backslash NP$: 1.64E-6 $S[dcl]\backslash NP[nb]$: 7.41E-17 $(S[dcl]\backslash NP)\backslash N$: 2.87E-17 ...	3
		X	$NP[nb]/N$: 0.05467 NP : 0.02439 $S/(S\backslash NP)$: 0.02124	4
			her	.

Table 4.3: Chart illustrating CI recovering a verb: Grey boxes are filled bottom-up by the parser; white boxes top-down by the learner.

1	2	3	4	5	
NP/N : 0.08428	$NP[nb]$: 0.00138	$(S[dcl]\backslash NP)/NP$: 3.38E-10 $S[dcl]/S[b]$: 2.77E-14 $S[dcl]/NP[nb]$: 1.44E-14 ...	$S[dcl]\backslash NP$: 6.22E-7 $S[dcl]/NP$: 6.22E-12 $S[dcl]/N$: 3.05E-13	$S[dcl]$: 1.0	1
The	N : 0.01890 NP : 0.00174 $S/(S\backslash NP)$: 0.00152	$(S[dcl]\backslash NP)/NP$: 2.48E-9 $((S[b]\backslash NP)/(S[b]\backslash NP))/NP$: 1.17E-10 $((S[dcl]\backslash NP)/(S[b]\backslash NP))/NP$: 1.06E-10 ...	$(S[dcl]\backslash NP)/(S[b]\backslash NP)$: 6.16E-7 $(S[dcl]\backslash NP)/(S[to]\backslash NP)$: 6.41E-9 $(S[b]\backslash NP)/(S[b]\backslash NP)$: 6.41E-9 ...	$S[dcl]\backslash(NP/N)$: 1.00E-4	2
	cat	$(S[dcl]\backslash NP)/NP$: 4.05E-11 $((S[b]\backslash NP)/(S[b]\backslash NP))/NP$: 1.91E-12 $((S[dcl]\backslash NP)/(S[b]\backslash NP))/NP$: 1.73E-12 ...	$(S[dcl]\backslash NP)/(S[to]\backslash NP)$: 1.05E-10 $S[b]\backslash NP)/(S[b]\backslash NP)$: 1.05E-10 $S[dcl]\backslash NP)/NP$: 4.73 ...	$S[dcl]\backslash NP$: 1.64E-6 $(S[dcl]\backslash NP)\backslash N$: 2.87E-17 $(S[dcl]\backslash NP)\backslash NP$: 2.65E-18	3
		X	$NP[nb]/N$: 0.05467 NP : 0.02439 $S/(S\backslash NP)$: 0.02124	$S[b]$: 5.14E-4 $NP[nb]$: 2.68E-4 $(S[b]/PP)/PP$: 1.06E-11 $S[b]/N$: 1.71E-12	4
			her	$((S[b]\backslash NP)/PP)/PP$: 0.50000 $S[b]\backslash NP$: 0.03125 N : 0.00567 NP : 0.00052 $S/(S\backslash NP)$: 0.00046	5
				ride	.

to handle large-scale language problems, but has the advantage over BF that it can invent new category types. CI can invent new category types, and it can operate over a larger number of instances than the RB system. It also has the benefit of being probabilistic and using the available labelled data in the form of a model from a supervised baseline parser to guide the search and rank results.

In the remaining chapters, we will investigate the efficacy of CI in addressing several word-learning tasks of increasing complexity.

Chapter 5

Convergence: A Toy Corpus

5.1 Introduction

The first half of this thesis has introduced Chart Inference as a novel method of deducing the CCG categories of unknown words in a sentential context. It has also described two baseline systems with which to compare it. In the remainder of the thesis, the three systems will be compared in the context of a range of practical tasks, beginning with a straightforward constructed corpus, and increasing in complexity with each successive chapter.

In each of these experiments we will define the corpora in use (including the seed lexicon, the training set, and the test corpus), the parameters under which CI is being employed, the evaluation method that best suits the task, and the relevant baseline against which to compare the results.

Our first avenue of inquiry is to evaluate the performance of CI and the two baseline systems over a manageable, artificial corpus. This chapter begins with the specification of the grammar and lexicon of the Toy Corpus, then employs it in an experiment with three initial seed settings. The first setting shows that all three learning algorithms are capable of converging on the target lexicon if provided with a sufficient seed, and the remaining two show that CI outperforms the baseline systems when initialised with smaller seed lexicons.

The small corpus allows for an online learning setting, where the initial seed lexicon is extended with each successive sentence, and the lexicon dynamically recalculated. We are able to examine the lexicon at any point, and can measure its similarity to the target lexicon to track improvement over time.

5.2 Corpus

To facilitate direct comparison the three learning methods, we use the Corpus 1 evaluation setting from Watkinson and Manandhar [1999a], which consists of a target lexicon and a uniformly-weighted PCFG language model used to randomly generate 1000 sentences[1]. In order to simply the learning task, we have standardised all tokens to lower case. The target lexicon, doubling as a unigram language model, has a uniform distribution. It contains 40 word-category pairs, including the full stop ($S\backslash S$), which was not in Watkinson's experiment, and one example of noun-verb ambiguity (*saw*) that was. The sentence-final punctuation has the standard category given by CCGbank, rather than $S\backslash S$, but that is immaterial because it is stripped from the CI learning process after it is used to determine the sentential result type (in this case, always $S[dcl]$). Figure 5.1 specifies the target lexicon in full and figure 5.2 shows the uniformly weighted PCFG model from which the training sentences (figure 5.3) are generated.

We specify three initial seed lexicons that present a range of challenges to the learner; they are shown in figure 5.4. Seed A is the most permissive setting; it contains one example of each of the category types present in the target lexicon. The lexical probability $p(W|C)$ for each entry is set to 1.0. Seed B is identical to Seed A, except that it lacks all mention of the ditransitive category type $((S[dcl]\backslash NP)/NP)/NP$. Seed C is the most challenging; it contains perfect knowledge of the determiners of the target lexicon and their lexical probabilities, as well as one noun to make learning possible[2]. The corpus resources are summarised below, a practice that will be duplicated for all subsequent experiments:

SEED Each of the three in figure 5.4.

TRAIN 1000 sentences automatically generated from target lexicon in figure 5.1 and PCFG in figure 5.2, a snippet of which is shown in figure 5.3.

TEST 40-item CCG target lexicon and model in figure 5.1.

[1]The full corpus was not included in any of Watkinson's papers, but its properties were outlined to such an extent that it was straightforward to recreate, though the reconstruction may differ slightly from the original in the distribution of category types.

[2]There are no sentences in the training corpus that consist of a determiner and just one other token.

Word	Category	$p(W\|C)$	Word	Category	$p(W\|C)$
john	NP	0.166	gave	$((S[dcl]\backslash NP)/NP)/NP$	0.2
mary	NP	0.166	called	$((S[dcl]\backslash NP)/NP)/NP$	0.2
jim	NP	0.166	sent	$((S[dcl]\backslash NP)/NP)/NP$	0.2
ellen	NP	0.166	threw	$((S[dcl]\backslash NP)/NP)/NP$	0.2
steve	NP	0.166	branded	$((S[dcl]\backslash NP)/NP)/NP$	0.2
anne	NP	0.166	a	NP/N	0.33
boy	N	0.11	some	NP/N	0.33
girl	N	0.11	the	NP/N	0.33
dog	N	0.11	smaller	N/N	0.166
fish	N	0.11	big	N/N	0.166
desk	N	0.11	bigger	N/N	0.166
elephant	N	0.11	hungry	N/N	0.166
statue	N	0.11	ugly	N/N	0.166
computer	N	0.11	small	N/N	0.166
saw	N	0.11	kissed	$(S[dcl]\backslash NP)/NP$	0.166
ran	$S[dcl]\backslash NP$	0.25	saw	$(S[dcl]\backslash NP)/NP$	0.166
walked	$S[dcl]\backslash NP$	0.25	kicked	$(S[dcl]\backslash NP)/NP$	0.166
coughed	$S[dcl]\backslash NP$	0.25	admired	$(S[dcl]\backslash NP)/NP$	0.166
blinked	$S[dcl]\backslash NP$	0.25	punished	$(S[dcl]\backslash NP)/NP$	0.166
.	.	1.0	timed	$(S[dcl]\backslash NP)/NP$	0.166

Figure 5.1: Toy Corpus target lexicon.

S	$\rightarrow$	NP VP	VP	$\rightarrow$	Vbar
Vbar	$\rightarrow$	IV	Vbar	$\rightarrow$	TV NP
Vbar	$\rightarrow$	DV NP NP	NP	$\rightarrow$	PN
NP	$\rightarrow$	Nbar	Nbar	$\rightarrow$	Det N
N	$\rightarrow$	Adj N			

Figure 5.2: CFG for generating the toy corpus.

```
1. anne admired a elephant .
2. a fish branded john ellen .
3. a small desk kissed a statue .
4. mary admired a fish .
5. john admired steve .
6. steve punished some girl .
7. the computer saw a boy .
8. some statue threw some saw anne .
9. a boy punished anne .
10. jim timed mary .
```

Figure 5.3: First ten sentences of the auto-generated toy corpus.

Seed A	john$\|NP$	1.0
One word of each category type	boy$\|N$	1.0
$\|A\| = 8$	ran$\|S[dcl]\backslash NP$	1.0
$cos(A,T) = 0.55$	timed$\|(S[dcl]\backslash NP)/NP$	1.0
	gave$\|((S[dcl]\backslash NP)/NP)/NP$	1.0
	the$\|NP/N$	1.0
	small$\|N/N$	1.0
	.$\|$.	1.0
Seed B	john$\|NP$	1.0
One word of every type except ditransitive	boy$\|N$	1.0
$\|B\| = 7$	ran$\|S[dcl]\backslash NP$	1.0
$cos(B,T) = 0.54$	timed$\|(S[dcl]\backslash NP)/NP$	1.0
	the$\|NP/N$	1.0
	small$\|N/N$	1.0
	.$\|$.	1.0
Seed C	the$\|NP/N$	0.33
Full set of determiners and one noun	some$\|NP/N$	0.33
$\|C\| = 5$	a$\|NP/N$	0.33
$cos(C,T) = 0.61$	boy$\|N$	1.0
	.$\|$.	1.0

Figure 5.4: Seed lexicons A, B, and C, subsets of the toy corpus.

Condition 1: Learning a new word:
If a sentence S of length N contains exactly $N-1$ known words:

Learn(S,W)
Find the highest-probability parse P_S
Update the lexicon with the word/category pairs from P_S

Condition 2: Learning a new category for a known word:
If a sentence S of length N contains exactly N known words
AND the parser cannot find a valid parse:

For each word $W_1 ... W_N$ in S:
Learn(S,W_i)
Find the highest-probability parse P_{W_i}
Find the highest-probability parse P_S in $P_{W_1 ... W_N}$
Update the lexicon with the word/category pairs from P_S

Figure 5.5: Learning setting for handling ambiguity, used in this chapter only.

5.3 Methods

Each learner [CI, BF, RB], is run in an iterative learning setting that recalculates the lexicon after every sentence seen. The setting is capable of learning multiple types for each word, since when it encounters a sentence for which no parse is possible, despite having each of the component words in the lexicon, it reconsiders each word in turn for the word/category pair that yields the highest-probability parse. Each learner plugs in seamlessly to this framework, which makes each capable of handling the ambiguity of the word 'saw' in the target lexicon. In addition to the new target word/category pair, all the lexical items used in the highest-probability parse are incremented in the lexicon, enabling the probability model to improve at each time step, and to keep adjusting lexical probabilities after all the word/category pairs have been learned.

The learning setting, specified in figure 5.5, handles one sentence at a time, deciding based on the number of known words in the sentence whether to use Condition 1 or 2, or skip it entirely. In Condition 1, it constructs a temporary hypothesis lexicon that contains all entries of the original lexicon, plus one entry each that pairs the OOL word with every category generated by the learner. Where the learner specifies a probability (CI only), these are integrated into the hypothesis lexical mode, but otherwise the prob-

abilities are considered uniform with respect to the word type. The parser then uses this temporary lexicon to parse the sentence. The highest-probability parse is broken down into its lexical entries and each of these is incremented in the working lexicon, including the new entry, for which the prior count would have been 0.

The second learning condition addresses cases of known words with unknown categories. It executes the same process as above, but once for each of the N words in the sentence, treating each in turn as the OOL item. This time, the highest-probability parse over the whole set is used to update the lexical counts. This method of addressing lexical ambiguity can be appended to any of the following algorithms, but is intractable for long sentences. Condition 2 is used only for the toy corpus in this chapter, where the search space over short sentences allows for new categories to be learned for known words.

The training corpus is traversed only once, not returning to skipped sentences, but the small size of the target lexicon means that in practice 300 sentences are sufficient for learning. We run all the learning methods over 1000 training sentences just to be safe. We use the StatOpenCCG parser is used for the bottom-up learning phases and the RASP tools [Briscoe et al. 2006] for preparing the input. The training sentences have been randomly generated before training time, and are always presented to the learners in the same order.

5.3.1 Baselines

The two learning systems described in chapter 3 provide a comparison for CI. The relevant baseline figures with which to compare our convergence experiments are Watkinson et al's reported findings that their method yields 100% parse accuracy when trained on a corpus of 500 automatically generated sentences. Since their learner is based on storing and reparsing each sentence as the lexicon expands, and ours passes over sentences that are not parseable at the moment, the number of sentences needed for lexical convergence is less relevant than the shape of the resultant learning curve. Watkinson and Manandhar [1999a] showed their system to be fully convergent (cosine similarity between the seed lexicon ($\vec{S}$) and the target lexicon ($\vec{T}$) above 0.99), whenever the seed lexicon contained at least one instance of each of the category types in the target lexicon, but here we test it with a variety of seed settings.

$\vec{x}$		$\vec{y}$	
John|NP	0.5	John|NP	1.0
Mary|NP	0.5		
		Mary|$(S[dcl]\backslash NP)/NP$	0.5
kissed|$(S[dcl]\backslash NP)/NP$	0.5	kissed|$(S[dcl]\backslash NP)/NP$	0.25
admired|$(S[dcl]\backslash NP)/NP$	0.5	admired|$(S[dcl]\backslash NP)/NP$	0.25

$$\vec{x}\cdot\vec{y} = (0.5*1.0)+(0.5*0)+(0*0.5)+(0.5*0.25)+(0.5*0.25) = 0.75$$
$$|\vec{x}| = \sqrt{0.5^2+0.5^2+0.5^2+0.5^2} = 1$$
$$|\vec{y}| = \sqrt{1.0^2+0.5^2+0.25^2+0.25^2} = 1.17$$
$$\cos(\vec{x},\vec{y}) = 0.75/(1*1.17) = 0.64$$

Figure 5.6: Sample calculation of cosine similarity between a target lexicon $\vec{x}$ and learned lexicon $\vec{y}$.

5.3.2 Evaluation

Watkinson and Manandhar [1999a] provides an evaluation metric that directly compares a seed or learned lexicon to a fixed target. Both lexicons to be compared are first sorted, then treated as feature vectors. Each word/category pair is interpreted as a dimension, and the associated probability a distance in that dimension. Entries in the target lexicon that are missing from the learned lexicon are treated as zeros, entries in the learned lexicon that are missing from the target lexicon likewise.

The degree of similarity between the two feature vectors is measured as the cosine of the angle between them. Equation 5.1 is the formula for calculating the cosine between two vectors [Manning and Schütze 1999]. An example calculation for two short lexicons is given in figure 5.6. For the purposes of this experiment, convergence is determined to have been reached when the cosine between the learned and target lexicons is greater than 0.99.

$$\cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{5.1}$$

The cosine method of comparing lexicons is only useful when the lexicons are fairly small, and there are not too many zeros in either direction. It gives a single score to an entire lexicon, which enables us to track the quality of a lexicon over time,

and evaluate different learned lexicons against each other. Cosine similarity is only a measure of lexicon matching, and not of usefulness in parsing, so a separate parsing evaluation is also performed.

For this chapter we also report lexical accuracy, which is strictly the number of word/category pairs in the learned lexicon that are correct, divided by the total number of learned lexical entries. This does not take into account the lexical probabilities. In the normal cases where the target lexicon is unknown, this calculation must be performed manually or with reference to existing linguistic resources, but in this case where the full toy lexicon is specified, lexical accuracy is simply the number of word/category pairs from the target lexicon that are present in the learned lexicon. Examining a lexicon's similarity to the target as well as its lexical accuracy allows for a full comparison of our three learning systems.

5.4 Results

When run over the toy corpus starting with Seed A, all three learners converge to the target lexicon in an identical sigmoid learning curve, as shown in figure 5.7. Because each learner sees the same 1000 sentences in the same order, the curves are identical. If the experiments had been run several times with the order of the corpus randomised, then averaged, the curves would be distinguishable.

In this case, all the necessary category types are known and the learners start from a position of having quite a large lexicon from which to expand. The lexical inventory goes from the original eight to the target 40 in 300 sentences, and achieves a cosine of .99 at sentence 388. There is a small initial plateau, since the learners have to skip the first ten sentences because they do not offer an N-1 learning opportunity. The curve is fairly stable, with only small decreases corresponding to discrepancies in the lexical probability model rather than incorrect word/category pairs.

This behaviour is what we expect, given Watkinson's description of lexical convergence. It also shows that in the simplest case, the CI, RB, and BF algorithms are equivalent.

When primed with Seed B, both the RB and CI algorithms again converge to the target lexicon in an identical manner. Figure 5.8 shows the learning curve for the three methods when the seed lexicon contains no mention of the ditransitive category type $((S\backslash NP)/NP)/NP$. Both RB and CI converge identically as expected, but BF, the lower curve, cannot learn any category types that are not attested in the seed, so it

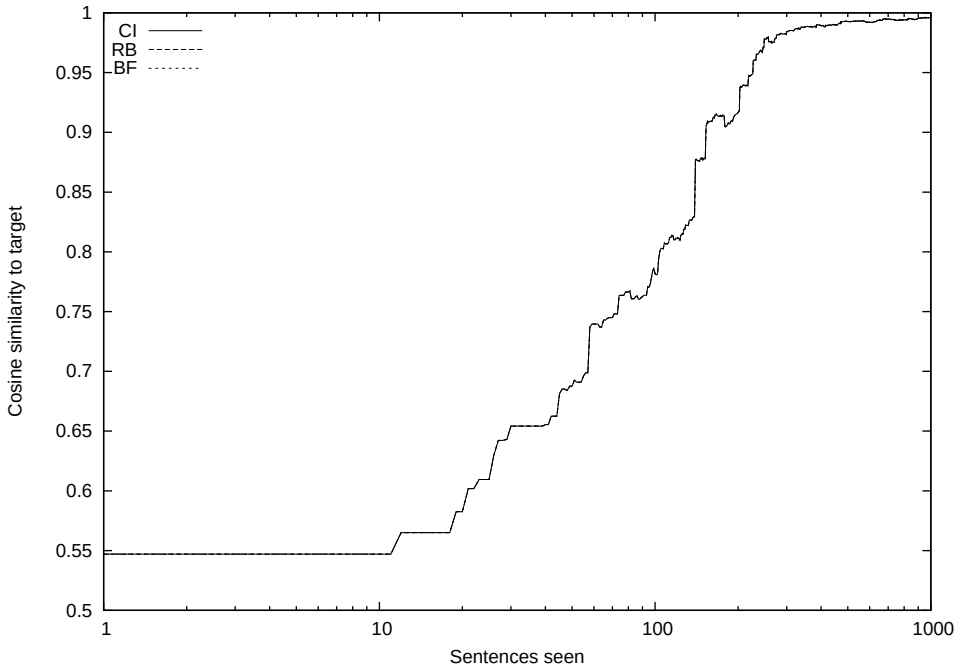


Figure 5.7: Learning curve for all three methods with Seed A. All methods converge within 1000 sentences with identical behaviour.

plateaus at 95% similarity.

The similarity between the two cosine curves in figure 5.8 is again due to the same sentences being seen in the same order, but now we can justify using this methodology, rather than averaging over multiple runs. The two curves are identical up to sentence 59, where CI and RB experience a decline and BF continues upward. This corresponds to the first encounter with a ditransitive (*jim called john mary* .). The two more flexible methods are able to learn this new category type, and show the dip associated with a 1.0 probability of a correct singleton, reflecting a shortcoming of the cosine similarity metric rather than of the learned lexicon. At the same crucial point, BF ignores this sentence since it cannot find a valid parse, and continues without the ditransitive, learning all the rest of the category types in a corresponding, but shallower, curve. CI and RB reach convergence at sentence 389, needing only one more sentence than with Seed A. BF levels off at 0.95, never having access to the ditransitive category.

When the seed is reduced to only three determiners and a noun, CI can still learn the complete lexicon, despite some initial missteps and a steeper curve. However, the other two methods have catastrophic failures (figure 5.9). BF never gets going, since it can only correctly learn the remaining nouns. RB is partially successful, peaking

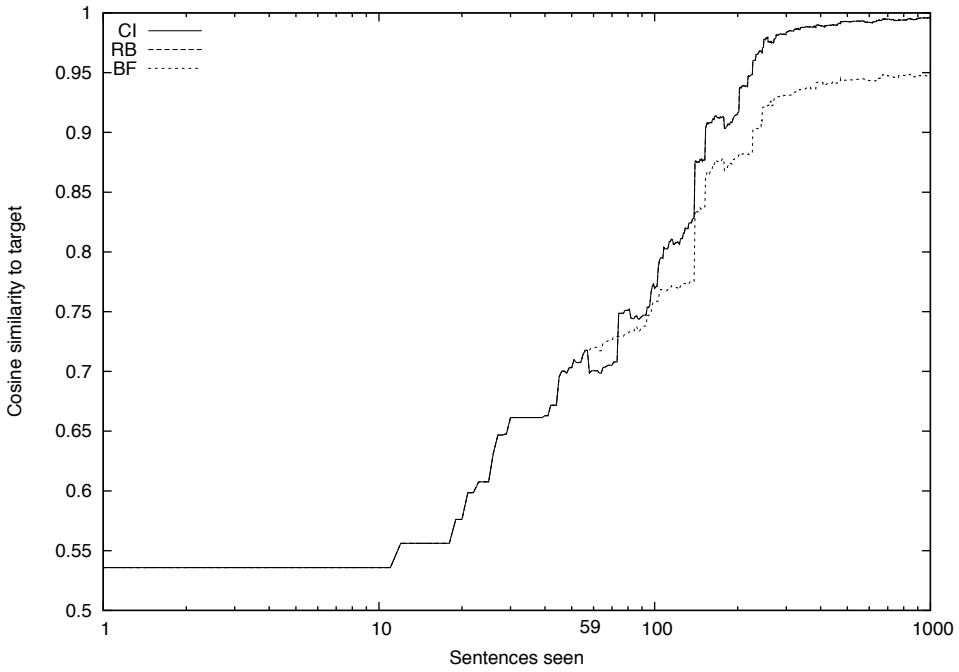


Figure 5.8: Learning curve for all three methods with Seed B. CI and RB converge identically, while BF differs.

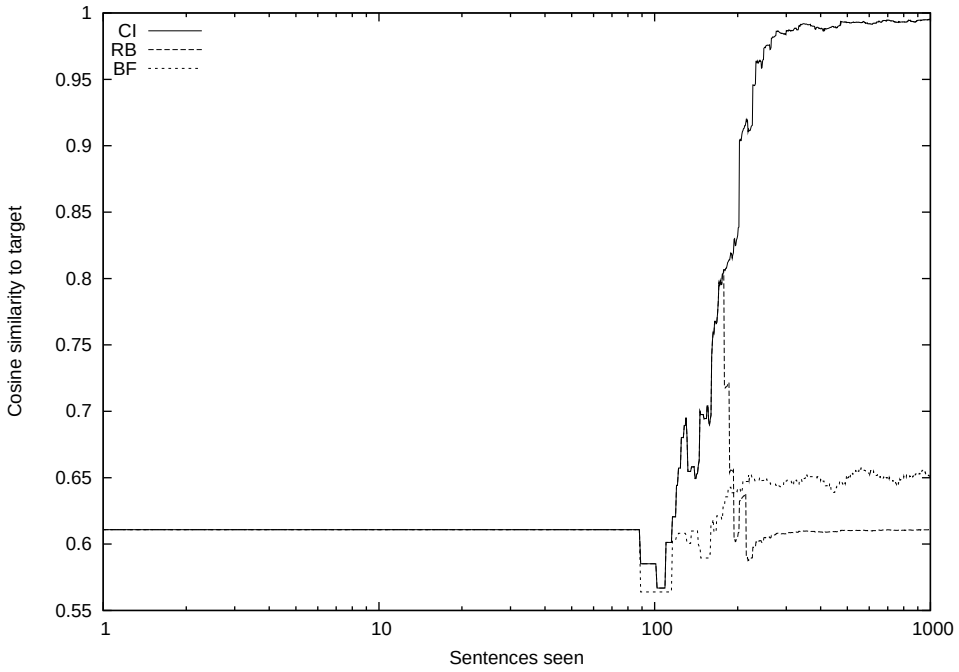


Figure 5.9: Learning curve for all three methods with Seed C.

Seed	BF	RB	CI
A	40/40 (100%)	40/40 (100%)	40/40 (100%)
B	35/40 (88%)	40/40 (100%)	40/40 (100%)
C	9/28 (32%)	40/44 (91%)	40/40 (100%)

Table 5.1: Lexical accuracy for the three learning methods, given three seeds.

at 80% similarity, but is thwarted by a bad decision around sentence 200 that quickly compounds to diverge from the target lexicon, ending up with higher coverage in the form of more lexical entries, but lower precision, as the final similarity plateaus at the same level as the original seed.

Only CI achieves full lexical convergence, but at sentence 334, 54 sentences fewer than needed for Seed A. It also has a longer initial plateau, as it has to reject more sentences before it can find one with N-1 known words. When it does begin to learn, the curve is steeper, showing that it is the quality of the seed, not just its size, that shapes the learning curve. A smaller seed is also sensitive to decreases in cosine as new words are learned, even when they are learned correctly. The two major dips in the CI curve in figure 5.9 correspond to two new lexical entries being learned correctly, but with hitherto unknown category types. As such, they have probabilities of 1.0, which negatively affects the cosine similarity measure until other words of that type are learned and the probabilities come into line with the target model.

5.5 Analysis

In the analysis we examine the characteristic errors of the systems that fail to converge. In all cases of convergence, the learners attain the exact lexicon inventory as the target, as well as a close approximation of the lexical probabilities. When they fail, however, it is for two reasons: first, words of unknown categories are impossible to learn (BF only), and second, because they are forced to choose the best category that forms a parse, they wrongly assign known categories to new words. When dealing with seed B, both the CI and RB methods yield a perfect 40-item lexical inventory. BF, however, comes up with only 35. This is because it fails to divine any category for those five ditransitive verbs in the corpus. This is failure of the first type only, and occurs because no known category for those 5 words produces a parse at all, so each sentence in which they occur is simply skipped.

When faced with seed C, however, errors of the second type occur. The BF learner yields 28 lexical entries from the 1000 training sentences. Knowing only two category types, it has access only to two-word (plus punctuation) sentences. Upon seeing the sentence 'The boy ran .' and having the first two words in the seed lexicon, the learner is forced to choose some category for *ran*, and since the learner-internal parser is more complex than the naive PCFG used to generate the sentence, it finds the highest probability parse $NP/N\ N\ N$. Once it has access to this erroneous entry (and the corresponding ones for every intransitive verb), it can start to make compounded errors. In fact, the BF lexicon after sentence 1000 contains 28 entries, consisting of three determiners, five nouns and the end punctuation, which are correct, four intransitive verbs wrongly labeled N, six NPs wrongly labeled as determiners, four nouns wrongly labeled as end punctuation, and five other assorted errors. These are summarised in the confusion matrix in table 5.2.

		Learned		
		NP/N	N	.
Target	NP/N	**3**	-	-
	N	1	**5**	4
	NP	6	-	-
	N/N	1	1	-
	$S[dcl]\backslash NP$	2	4	-
	$(S[dcl]\backslash NP)/NP$	-	-	-
	$((S[dcl]\backslash NP)/NP)/NP$	-	-	-
	.	-	-	**1**

Table 5.2: Confusion matrix for BF with seed C.

The RB learner yields 44 lexical items from seed C: the 40 correct ones, and four additional spurious entries:

small$\|N/((S[dcl]\backslash NP)/NP)$	1.0
some$\|NP/((S[dcl]\backslash NP)/NP)$	1.0
threw$\|(NP/(NP/N))\backslash NP$	1.0
threw$\|(NP/(NP/((S[dcl]\backslash NP)/NP)))\backslash NP$	1.0

These arise from the ambiguity of the word *saw*. Once two entries for *saw* are known, it opens the way for a misparse. CI deals with this because it has a full sta-

tistical model and chooses the highest-probability parse, but RB prefers the simplest parse, and so it is open to this type of error. When it encounters the ambiguity problem with *saw*, it erroneously decides to give a spurious category type to another word in the sentence. This opens the lexicon up to compounded errors in the same way as BF above. The fact that these spurious category types have only been seen with one word type skews the cosine measurement and leads us to believe that the learned lexicon is worse than it is. While it has learned 40/44 entries correctly, Figure 5.9 implies that it has done worse than BF. This points out a weakness in using cosine similarity to assess lexicon quality. As such, we will dispense with it in the coming chapters and focus on measures more directly related to lexicon coverage and parsing performance.

5.6 Discussion

This experiment has shown that both Chart Inference and the Rule Based learner conform to an expected learning curve over a toy corpus in the same way as predicted by Watkinson's Brute Force method. However, the differences in the methods emerge when the initial seed lexicon is varied. Both BF and RB are forced to come up with a best guess in more situations than it is advisable. CI has the flexibility to fail to return an answer for sentences that do not contain enough information for an informative guess. The BF learner performs as expected; it converges on the target lexicon when all the necessary category types are available to it. It has no recourse, however, when some category types are unknown; it cannot hope to learn the entire lexicon, and makes incorrect inferences as a result. The RB learner can invent new category types, but it lacks a full statistical model and therefore has difficulty distinguishing between valid parses. CI is the most robust of the three, achieving lexical convergence where the others fail. It is capable both of learning new category types and of applying the full strength of the statistical parsing model and the expressiveness of CCG.

Real natural language experiments are projected to have lower results because model errors and missing categories will occur in the bottom-up partial parsing step. The next chapter will address the problem of natural language data.

Chapter 6

Coverage: The McGuffey Corpus

6.1 Introduction

This chapter is the first foray into testing our word learners on a true natural-language corpus. We focus on a small subset of English, as represented by a novel corpus of children's books, developed and annotated specifically for this thesis. We move away from the online learning setting of the previous chapter, and instead define an offline learning task, in which precision and recall can be measured across the whole test corpus.

The first experiment in this chapter sets the learners the task of recovering the correct category for each word in the corpus in isolation, while having access to the full statistical lexicon for the context words. The second moves on to the more complex task of learning completely unknown words in unseen contexts, given an imperfect seed lexicon, a task very similar to the intended use for CI. We conclude with a very controlled experiment exploring the capacity of the BF, RB, and CI learners to discover OOL category types.

6.2 Corpus

Because we seek to test our learner in an environment that is a small but valid subset of English, and that requires CCG category annotation and gold-standard parse trees, we have created a new corpus expressly for this purpose. Using a children's book as a source text affords the appropriate lexicon size. Since it is a whole book, rather than a collection of unrelated sentences, it ensures that the lexical inventory is low, but each lexical item has been seen multiple times. As such, the McGuffey corpus is

Corpus	MG1	CCGbank
Sentences	546	39604
Tokens	6024	929552
Types	749	44210
Category Types	196	1286
Lexical Entries	1153	74669
Tokens / Sentence	11.03	23.47
Cat Types / Word Type	1.54	1.69
Word Types / Cat Type	5.88	58.06

Table 6.1: Composition of McGuffey Volume 1 (MG1) vs. CCGbank §02-21.

a small natural language corpus, with possible future applications for modelling child language acquisition.

The source for this corpus is William Holmes McGuffey's Eclectic Readers [McGuffey 1836; 1837], first published in 1836 and subsequently much revised[1]. The readers comprise six progressive volumes, designed specifically to teach children to read and write. Much of the content is designed for what, at the time, was considered the moral education of children, including selections from the Bible, stories about being kind to animals, and admonitions of obedience and politeness. Volume 1 contains mostly short sentences, starting with 'the cat is on the mat' and getting gradually more complex until Volume 6, which contains selections from Dickens, Longfellow, and Coleridge, among others. Owing to the historical context of their production, many passages would be considered racist or offensive in modern usage. No effort has been made to update the content; may the user be warned. The corpus contains only the prose portion of the readers. Introductions, word lists, exercises, and instructions have been removed. A small portion of the first reader is poetry, but as it is nonstandard in language and use of line breaks, it is not included in this version of the corpus. Subsequent releases may have the poetry restored.

For this thesis we have undertaken to annotate Volumes 1 and 2; the former as a development set and the latter an evaluation set. Volume 1 of the McGuffey corpus (MG1) consists of 546 sentences that have been manually annotated with CCG cate-

[1]The raw text of William Holmes McGuffey's Eclectic Reader is available as an e-book from Project Gutenberg at http://www.gutenberg.org/ebooks/14640. The annotated corpus is available for download at http://homepages.inf.ed.ac.uk/s0672485/mcguffey.html.

gories, automatically parsed, and then corrected. MG1's sentences vary in length from 2-34, with a mean sentence length of 10.57, and a median of 8. Table 6.1 compares some relevant statistics of the MG1 corpus with those of CCGbank. The biggest difference between them, apart from size, is the lexical inventory, as represented by the number of word types per category type. The lexical complexity, or category types per word type, is very similar. This indicates that conclusions drawn from the very small McGuffey corpus are likely to scale to the state-of-the-art CCGbank corpus.

Volume 2 (MG2) comprises 801 sentences, annotated in the same manner as Volume 1, though not as reliably. The McGuffey corpus serves as a gentle testbed for development purposes. It is mainly composed of simple declarative sentences, but also touches on questions, quotations, passives, and other complex constructions. As it was designed as a tool to help children learn to read, the rate of introduction of new items is slower than newswire corpora.

The annotation of the McGuffey Corpus is not entirely compatible with that of CCGbank, as it was originally undertaken as a study of the treatment of punctuation in CCG. For example, commas, quotation marks and negative particles have different category types than CCGbank's, and adjectival predicates ($S[adj]\backslash NP$) are conflated with standard noun modifiers (N/N). As such, the McGuffey Corpus will not be directly helpful in building a better CCGbank, but it is a very useful as a small and internally consistent development corpus.

Figure 6.1 shows the first five sentences of MG1 in their raw machine-readable form. They are in CCGbank format. Each node of the tree is labelled either as a leaf (L) or non-terminal (T). Leaf nodes are annotated with their CCG category and Penn Treebank part of speech[2]. Non-terminals have a CCG category type and two numbers designating headedness and number of daughters.

6.3 Empirically Setting Learning Parameters

In this section we set the unary rule inventory and filter settings specific to the McGuffey corpus. We use these parameter settings for the experiments in this chapter, and they inform the choices made in subsequent experiments. Because we are now operating in a natural language setting, we must set some parameters to limit the complexity of the problem.

[2]Where it looks like information is duplicated within the leaf nodes, this is only to make the format compatible with the StatCCG parser.

```
ID=mg1.1 PARSER=GOLD NUMPARSE=1
(<T NP[nb] 0 2> (<T NP[nb] 0 2> (<L NP[nb]/N DET DET The NP[nb]/N>) (<L N
NN NN dog N>) ) (<L . . . . .>) )
ID=mg1.2 PARSER=GOLD NUMPARSE=1
(<T S[dcl] 0 2> (<T S[dcl] 1 2> (<T NP[nb] 0 2> (<L NP[nb]/N DET DET The
NP[nb]/N>) (<L N NN NN dog N>) ) (<L S[dcl]\NP VBD VBD ran S[dcl]\NP>) )
(<L . . . . .>) )
ID=mg1.3 PARSER=GOLD NUMPARSE=1
(<T NP[nb] 0 2> (<T NP[nb] 0 2> (<L NP[nb]/N DET DET The NP[nb]/N>) (<L N
NN NN cat N>) ) (<L . . . . .>) )
ID=mg1.4 PARSER=GOLD NUMPARSE=1
(<T NP[nb] 0 2> (<T NP[nb] 0 2> (<L NP[nb]/N DET DET The NP[nb]/N>) (<L N
NN NN mat N>) ) (<L . . . . .>) )
ID=mg1.5 PARSER=GOLD NUMPARSE=1
(<T S[q] 0 2> (<T S[q] 0 2> (<T S[q]/PP 0 2> (<L (S[q]/PP)/NP VBZ VBZ Is
(S[q]/PP)/NP>) (<T NP[nb] 0 2> (<L NP[nb]/N DET DET the NP[nb]/N>) (<L N
NN NN cat N>) ) ) (<T PP 0 2> (<L PP/NP IN IN on PP/NP>) (<T NP[nb] 0 2>
(<L NP[nb]/N DET DET the NP[nb]/N>) (<L N NN NN mat N>) ) ) ) (<L . . .
? .>) )
```

Figure 6.1: Corpus snippet showing the first five sentences of MG1.

Restriction	sents	words	UP	UR	LP	LR	T (m)
len ≤ 8	289	1517	85.96	80.69	68.12	63.94	11
len ≤ 9	327	1857	82.95	75.18	64.88	58.80	18
len ≤ 10	354	2106	81.13	71.65	63.87	56.41	25
-[,;:and]	257	1525	85.10	76.07	71.31	63.74	39
-[,;:and], len ≤ 10	235	1250	**87.52**	**80.8**	**73.74**	**68.08**	11

Table 6.2: Performance and head-dependency results for CI over MG1, comparing parameters of sentence length and complexity, with a minimal rule set.

The problem with unary rules in StatCCG is that the model has access to all the rules, regardless of frequency. That isn't a problem for the parser in Hockenmaier's original configuration, since it only parses using rules that are instantiated in the model. However, since we are using the same rules with no model restrictions and treating all unary rules as equal, the rare rules become overrepresented and make the search space unnecessarily large, increasing both processing time and overgeneration. To fix this, one option is to remove all the extraneous rules from the parser. In this condition, it is important to note that the same model is used for bottom-up and top-down parsing. Another option is to remove those rules from the top-down model only; doing so would necessitate keeping two separate models for the learner to use.

We performed a limited parameter evaluation experiment to attempt to establish how reducing the set of unary rules affects the performance and time elapsed for learning each word in context in MG1. Three different rule configurations were created by hand and tested for their effect on unlabelled precision and recall (UP and UR), labelled precision and recall (LP and LR) and time (T). To decide which rule set is optimal, we must weigh precision against efficiency.

In the course of this experiment, it became clear that the majority of the time is being spent on a small number of sentences. When we examine these sentences, we find that they are long and complex. We therefore expand our search space to include a second parameter, and attempt to examine the effect of putting length or complexity restrictions on the sentences that the learner tries to learn from.

Internal punctuation and coordination are the two phenomena that most easily estimate the complexity of a sentence. The length of a sentence is correlated with the number of commas it contains. Any other restriction on complexity, like disqualifying the word *that*, because it sometimes introduces relative clauses, would result in a bias

Restriction	sents	words	UP	UR	LP	LR	T (m)
len $\leq$ 8	289	1517	90.22	85.76	69.83	66.38	16
len $\leq$ 9	327	1857	88.31	81.37	67.33	62.04	29
len $\leq$ 10	354	2106	86.41	77.92	66.30	59.78	49
-[,;:and]	257	1525	93.83	85.77	76.83	70.23	249
-[,;:and], len $\leq$ 10	235	1250	**94.48**	**88.96**	**77.15**	**72.64**	17

Table 6.3: Performance and head-dependency results for CI over MG1, comparing parameters of sentence length and complexity, with a small rule set.

Restriction	sents	words	UP	UR	LP	LR	T (m)
len $\leq$ 8	289	1517	89.90	85.63	69.62	66.32	18
len $\leq$ 9	327	1857	88.74	81.91	68.03	62.79	35
len $\leq$ 10	354	2106	87.5	79.11	66.65	60.26	57
-[,;:and]	257	1525	**96.71**	88.72	**78.70**	72.20	315
-[,;:and], len $\leq$ 10	235	1250	96.44	**91.12**	78.49	**74.16**	26

Table 6.4: Performance and head-dependency results for CI over MG1, comparing parameters of sentence length and complexity, with a full rule set.

against learning the category types that are the target of this exercise. For that reason, we limit the restrictions to a list of stopwords: [, ; : and], the presence of any of which cause the sentence to be skipped.

We ran the full parameter space and evaluated against the head-dependencies in MG1, as shown in figures 6.2, 6.3 and 6.4. The first trend that is apparent is that limiting the rule set lowers performance considerably. We can see that access to rare rules allows higher precision in the rare cases when those rules are necessary to form a complete parse, shown by the higher precision and recall scores in table 6.4, but in most cases this increase the search space and results in longer run times. Limiting the complexity without using a length restriction yields superior accuracy, but causes runtimes to explode, especially with larger rule sets. However, when both restrictions are in place, we see the best performance with reasonable efficiency, without the need to place artificial constraints on the set of unary rules.

Table 6.4 shows that the best configuration is the full rule set, restricted to sentences of length 10 and containing no internal punctuation or coordination. It yields the optimum combination of high precision and recall, within a reasonable space of time. We

Sentence #	Token #	Context	Target
1	1	**X** dog .	The:=$NP[nb]/N$
1	2	The **X** .	dog:=N
2	3	**X** cat .	The:=$NP[nb]/N$
2	4	The **X** .	cat:=N
⋮	⋮		
546	6024	... to jump to the **X** .	stone:=N

Table 6.5: The coverage experiments attempt to recover each token of the test set in turn; P and R are calculated over token matches.

will use this configuration throughout the rest of the paper, unless stated otherwise.

6.4 A Natural Language Experiment

In the first natural language learning task we investigate how well CI performs at recovering a wide range of category types in complex settings.

6.4.1 Methods

In the first experiment we traverse the MG1 corpus in one pass, attempting to learn each word token in turn and comparing the learned category set to the gold standard annotation. Because we know that the lexicon contains all the necessary entries to correctly parse all the sentences, this equates to an assessment of the lexical coverage power of the learning algorithm.

Each word token is tested in isolation, so we report precision and recall over all words in the test set, as compared to the gold-standard labels. There are 546 sentences in MG1, and 6024 tokens, so performance is measured out of a total of 6024 matches. This is summarised in table 6.5.

Figure 6.2 outlines the process of producing new parsed sentences out of raw text. The process begins like the previous experiment, but then the category set generated by the learner is passed back to the parser, so it can incorporate this new information into its lexicon and produce a full parse. The Hypothesis lexicon is cleared after every sentence.

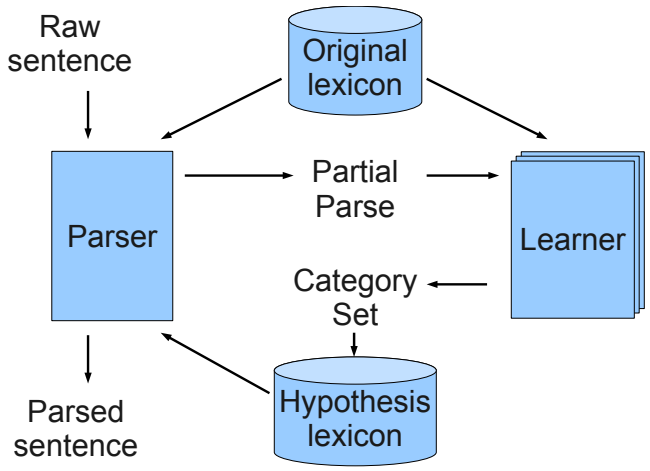


Figure 6.2: Learning framework for coverage experiments in chapter 7.

6.4.1.1 Parameter Settings

The experimental setting for this experiment is one stage less than the convergence experiments in the previous chapter. The parser is trained on MG1, and then tested on the same corpus, treating one word at a time as unknown as in table 6.5. There is not a separate training and test phase, and the lexicon is never changed. We test each sentence afresh in isolation from the rest, to evaluate the efficacy of CI on the sentence level.

SEED Gold standard MG1 lexicon, with target word blacked out.

TEST Gold standard MG1 CCG category labels.

6.4.1.2 Baseline

We continue using the BF and RB learners for comparison, and employ a naive POS baseline, generated by simply tagging each target word with the category that is most frequently associated with its POS tag. No context is taken into account. The POS tags have been incorporated into the parsing model. All POS tags used are gold standard labelled by hand; automatic tagging is projected to yield slightly worse results.

6.4.1.3 Evaluation

We report precision, recall, and F-score figures against the gold-standard category labels for each word token in the corpus, referred to subsequently as category match scores. This differs from standard parse evaluation, as the internal structure of the

	Best Guess			In Category Set			
	P	R	F	P	R	F	Time (m)
POS	63.71	63.71	63.71	*	*	*	1
BF	**80.65**	69.36	74.58	*	*	*	1825
RB	39.77	37.92	38.82	68.46	65.28	66.83	12
CI	78.63	**74.16**	**76.33**	**97.03**	**91.52**	**94.20**	22

Table 6.6: Category match results for the three systems on the McGuffey corpus, training and testing on MG1, compared to a naive POS baseline. *Best Guess* evaluates over the single highest-scoring result, *In Category Set* over the whole list of possible categories returned. POS and BF contain (*) for the latter because they always return the full set of possible categories.

parse is not taken into account. Where appropriate, learner speed is also reported in terms of time needed to traverse the test set.

6.4.2 Results

Table 6.6 compares the category match accuracy across the three systems, as well as the baseline that chose the most probable category for the target word's POS. Two tasks are scored: *Best Guess*, where we evaluate the single highest-scoring category against the gold-standard tag, and *In Category Set*, where we check to see if the gold tag is present somewhere in the category set returned by the learner. The POS and BF methods get perfect scores in the latter by definition, since their category sets consist of all known categories, and all categories seen with the target word's POS tag, respectively. However, these figures are left out of table 6.6 so as not to mislead the reader.

BF gets a respectable F-score in the *Best Guess* task, easily outdoing the POS baseline, but takes 30 hours to do so under our computing configuration[3], since it is searching over all possible categories. RB is markedly worse in performance, but also remarkably fast. CI combines the merits of both BF and RB, yielding a higher F-score than BF and a processing time similar to RB.

On the *In Category Set* task, we allow for the presence of the correct category type in the Top Ten list to constitute success. This yields a dramatic increase in F-scores for both the RB and CI learners. RB's performance is nearly doubled, but it is still

[3] 3GHz dual core Intel Xeon with 8GB RAM

	P	R	F
Top 1	78.63	74.16	76.33
Top 2	85.92	81.04	83.41
Top 3	91.18	86.00	88.51
Top 5	95.42	90.00	92.63
Top 10	97.03	91.52	94.20

Table 6.7: Rank evaluation of category sets returned by CI on MG1, when the parameter of category set size is adjusted from 1 to 10.

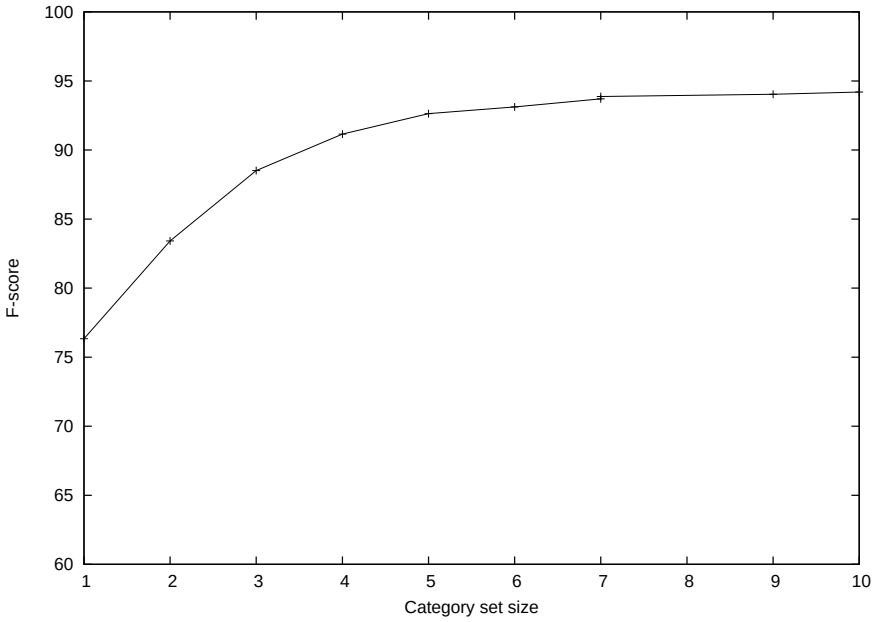


Figure 6.3: F-score of CI over category matches on MG1 as category set size is increased.

outperformed by CI, which reaches 97% precision and 92% recall. The original goal of focusing on precision over recall is reflected here.

Of all the methods, CI is the only one that returns a proper ranked category set. Where the correct category is present in the set, the average rank is 1.4. An F-score of 76.33 is achieved by just picking the top ranking category, but that goes up quickly if we allow the answer to be drawn further down the list. The default size for a returned category set is 10. Table 6.7 shows how precision, recall and F-score for the CI learner are affected by widening the net. Figure 6.3 shows that the F-score grows quickly at first, then asymptotically as the allowed set size reaches the default 10.

While the rule-based system is quick, it performs poorly on both precision and

recall. Its poor performance in the *Best Guess* task is due to the fact that it uses no statistics to choose the best category set or parse, opting instead to return the *simplest* category, that is, the one (or more) with the lowest valency. Here we have been generous, allowing any correct category of the lowest valency to count as a success.

CI outperforms the baseline methods on a natural language corpus. It is robust and does not depend on knowing all the categories ahead of time, or on having four or fewer discernible constituents. For the BF learner to compete, it would have to have access to a much faster parser. For the RB learner to stand up to either of these, it would need to employ a more robust system of ranking candidate categories, which would make it slower.

6.4.3 Analysis

In this section we examine the output for CI in depth, to investigate whether there are any systematic failures in the algorithm that would prevent it from performing adequately on more complex natural-language tasks. Table 6.8 breaks down the results by category type, reporting the number of times the type was correctly matched, versus the number of times it was returned as a guess. The table is sorted by the "Guessed" column, showing that the most common types tend to have the high precision values.

Table 6.8: By-category precision for CI over the McGuffey corpus.

Category	Correct	Guessed	Precision
NP	175	208	0.84
N	170	190	0.89
$NP[nb]/N$	130	166	0.78
PP/NP	75	102	0.74
N/N	63	81	0.78
$(S[dcl]\backslash NP)/(S[b]\backslash NP)$	27	27	1.0
$(S[b]\backslash NP)/NP$	22	27	0.81
$S[q]/S[b]$	22	27	0.81
$(S[dcl]\backslash NP)/NP$	21	25	0.84
$(S[dcl]\backslash NP)/PP$	17	24	0.71
$(S[dcl]\backslash NP)/(N/N)$	22	23	0.96
S/S	14	16	0.875
$(S[to]\backslash NP)/(S[b]\backslash NP)$	13	15	0.87
...			

Table 6.8: (continued)

Category	Correct	Guessed	Precision
$(S[dcl]\backslash NP)/(S[pt]\backslash NP)$	13	14	0.93
$(S[b]\backslash NP)/PP$	12	13	0.92
$S[b]\backslash NP$	7	11	0.64
neg	5	11	0.45
$(S\backslash NP)\backslash(S\backslash NP)$	1	11	0.09
$S[dcl]\backslash NP$	8	10	0.8
$(S[iq]/(S[dcl]/(N/N)))/(N/N)$	9	9	1.0
$(S[pt]\backslash NP)/NP$	8	8	1.0
$(S\backslash NP)/(S\backslash NP)$	3	7	0.43
$S[frg]/NP$	0	7	0
$(N/N)/(N/N)$	1	6	0.17
$(S[dcl]\backslash NP)/(S[ng]\backslash NP)$	5	5	1.0
$((S[dcl]\backslash NP)/PP)/NP$	5	5	1.0
$(S[b]\backslash NP)/S[dcl]$	4	5	0.8
$(S[b]\backslash NP)/(S[to]\backslash NP)$	4	4	1.0
$(S[pt]\backslash NP)/PP$	4	4	1.0
$(NP[nb]/N)\backslash NP$	3	4	0.75
$S[inv]/S[b]$	1	4	0.25
$S[imp]/NP$	0	4	0
$S[q]/S[pt]$	3	3	1.0
$(S[iq]/(S[dcl]/NP))/NP$	3	3	1.0
$(S[dcl]\backslash NP)/(PP/NP)$	3	3	1.0
$(S[b]\backslash NP)/(N/N)$	3	3	1.0
$(S[b]\backslash NP)/(PP/NP)$	3	3	1.0
$((S[b]\backslash NP)/(S[b]\backslash NP))/NP$	3	3	1.0
$((S[dcl]\backslash NP)/NP)/NP$	3	3	1.0
$(S[imp]/(S[b]\backslash NP))/NP$	2	3	0.67
$((S[b]\backslash NP)/PP)/NP$	1	3	0.33
$S[wq]/(S[inv]/NP)$	1	3	0.33
PP/PP	0	3	0
$S[tag]/NP$	0	3	0
$S[wq]/(S[dcl]\backslash NP)$	2	2	1.0
...			

Table 6.8: (continued)

Category	Correct	Guessed	Precision
$S\backslash S$	2	2	1.0
$S[imp]/S[iq]$	2	2	1.0
$(S[q]/NP)/NP$	2	2	1.0
$(S[q]/PP)/NP$	2	2	1.0
$(S[dcl]\backslash NP)/(S[to]\backslash NP)$	2	2	1.0
$(S[b]\backslash NP)/S[iq]$	2	2	1.0
$((S[pt]\backslash NP)/PP)/NP$	2	2	1.0
$((S[b]\backslash NP)/(PP/NP))/NP$	1	2	0.5
NP/NP	1	2	0.5
$(N/N)/PP$	1	2	0.5
$S[iq]/S[dcl]$	1	2	0.5
$((S\backslash NP)\backslash(S\backslash NP))/S[dcl]$	0	2	0
$S[iq]/(S[dcl]/NP)$	0	2	0
$S[iq]/NP$	0	2	0
$S[ng]\backslash NP$	0	2	0
$S[wq]/(S[inv]/PP)$	1	1	1.0
$S[pt]\backslash NP$	1	1	1.0
$S[inv]/NP$	1	1	1.0
$S[imp]/PP$	1	1	1.0
$S[imp]/S[imp]$	1	1	1.0
$PP/S[dcl]$	1	1	1.0
$(S[q]/(N/N))/NP$	1	1	1.0
$(S[pt]\backslash NP)/(S[ng]\backslash NP)$	1	1	1.0
$(S[imp]/(S[to]\backslash NP))/NP$	1	1	1.0
$(S[inv]/NP)/NP$	1	1	1.0
$(S[dcl]\backslash NP)/S[iq]$	1	1	1.0
$(PP/PP)/(N/N)$	1	1	1.0
$((S[b]\backslash NP)/(S[to]\backslash NP))/NP$	1	1	1.0
$((S[b]\backslash NP)/PP)/PP$	1	1	1.0
$((S[dcl]\backslash NP)/(PP/NP))/NP$	1	1	1.0
$((S[dcl]\backslash NP)/(S[b]\backslash NP))/NP$	1	1	1.0
$((S\backslash NP)\backslash(S\backslash NP))\backslash NP$	1	1	1.0
...			

Table 6.8: (continued)

Category	Correct	Guessed	Precision
$((S[ng]\backslash NP)/PP)/NP$	1	1	1.0
$((S[b]\backslash NP)/NP)/(PP/NP)$	0	1	0
$((S[b]\backslash NP)/PP)/(N/N)$	0	1	0
$((S[b]\backslash NP)/NP)/(PP/NP)$	0	1	0
$((S[b]\backslash NP)/PP)/(N/N)$	0	1	0
$((S[dcl]\backslash NP)/(S[to]\backslash NP))/(N/N)$	0	1	0
$((S[dcl]\backslash NP)/PP)/PP$	0	1	0
$((S[ng]\backslash NP)/NP)/(PP/NP)$	0	1	0
$((S[pt]\backslash NP)/NP)/(PP/NP)$	0	1	0
$((S\backslash NP)\backslash(S\backslash NP))/((S\backslash NP)\backslash(S\backslash NP))$	0	1	0
$((S\backslash NP)\backslash(S\backslash NP))/PP$	0	1	0
$(S[b]\backslash NP)/S[b]$	0	1	0
$(S[ng]\backslash NP)/NP$	0	1	0
$(S[ng]\backslash NP)/PP$	0	1	0
$(S[q]/S[iq])/NP$	0	1	0
$(S[to]\backslash NP)/((S[b]\backslash NP)/NP)$	0	1	0
$(S\backslash S)/(S\backslash S)$	0	1	0
$(S\backslash S)/NP$	0	1	0
$(S\backslash S)\backslash NP$	0	1	0
NP/PP	0	1	0
$NP\backslash NP$	0	1	0
$S[frg]$	0	1	0
$S[frg]/(N/N)$	0	1	0
$S[imp]$	0	1	0
$S[imp]/S[b]$	0	1	0

For the 102 category types attempted, 32 are never tagged correctly by CI. Most of these are only attempted once, so we don't have enough evidence to attribute the blame to the learner or to the sentence. For any category type that was attempted more than once and never with success, we examine all the instances by hand. Analysis shows that all of these errors are attributable to the model, and do not represent flaws in the learning algorithm itself.

We classify the errors into four types below. It is important to note that in all of these cases, the *returned* category is the one assigned the highest probability by the learner, and that the correct category is present lower down in the category set[4].

6.4.3.1 Sentence Type Errors

The most common type of error has the structure of the category correct, but chooses the wrong sentential type for the result.

	Sentence	Gold Standard	Best Guess
1	**O** Ben ! (x5)	$S[frg]/NP$	$S[iq]/NP$
2	**O** mother ! (x2)	$S[frg]/NP$	$NP[nb]/N$
3	**See** Rab !	$S[imp]/NP$	$S[iq]/NP$
4	**See** Ann !	$S[imp]/NP$	$S[iq]/NP$
5	**What** a bright June day !	$S[iq]/NP$	$S[imp]/NP$
6	Oh ! **Thank** you .	$S[tag]/NP$	$S[frg]/NP$
7	**Thank** you very much .	$S[tag]/NP$	$(S\backslash S)/NP$

In (1), which occurs five times in the error set with different NPs, the learner is correct in deducing that the target takes an NP to the right, but chooses to treat the whole sentence as an indirect question, rather than a sentence fragment. On the whole, this is unlikely to cause problems, since exclamations of this type could be analysed in several ways. A similar process is at work in (2), where *O* is deemed a determiner, since that pattern is common in the corpus and there is no evidence to contradict it. (3) and (4) are problems, since parsing a sentence as an indirect question rather than an imperative would have negative effects were these parses to be used in certain types of systems. (5) is the same behaviour in reverse. (6) and (7) are interpreted as a fragment and a sentential modifier, respectively, rather than tag question, which causes no serious concern.

6.4.3.2 Argument/Adjunct Confusion

The second type of error sees adjuncts interpreted as arguments.

[4]With a few exceptions, which will be discussed in second half of the error analysis.

	Sentence	Gold Standard	Best Guess
1	**See** the eggs in the nest !	$S[imp]/NP$	$(S[imp]/PP)/NP$
2	**See** the duck on the pond !	$S[imp]/NP$	$(S[imp]/PP)/NP$
3	Sweet little birds were **singing** all around her .	$S[ng]\backslash NP$	$(S[ng]\backslash NP)/PP$
4	**What** a big room for such a small school !	$S[iq]/NP$	$(S[imp]/PP)/NP$

In these four cases, the result type and one argument are correct, but there is an additional prepositional argument that is canonically an adjunct. (4) is arguably correct, and possibly a better analysis than the gold-standard annotation.

6.4.3.3 Modifier Errors

The learner has to make decisions about modifier attachment based on the model. In four cases it chose incorrectly.

	Sentence	Gold Standard	Best Guess
1	Sweet little birds were singing **all** around her .	PP/PP	$(S\backslash NP)\backslash(S\backslash NP)$
2	Look **down** into this bush .	PP/PP	$S\backslash S$
3	She wears glasses **when** she reads .	$((S\backslash NP)\backslash(S\backslash NP))/S[dcl]$	$(NP\backslash NP)/S[dcl]$
4	No one could get them **while** Rab was there .	$((S\backslash NP)\backslash(S\backslash NP))/S[dcl]$	$(NP\backslash NP)/S[dcl]$

(1) attaches to the verb phrase to the left, rather than the prepositional phrase to the right. The same is true for (2), which chooses to attach to the S[imp] to the left. (3) and (4) exhibit similar behaviour, getting the direction of the attachment right, but the scope wrong. They are erroneously analysed as relative pronouns, which is a more common occurrence in the model than the correct adverbial analysis.

6.4.3.4 Remaining Errors

The remaining five errors are either combinations of the previous kinds, or ambiguous analyses.

	Sentence	Gold Standard	Best Guess
1	Ca n't you get **back** to the hen ?	PP/PP	NP
2	Ben White is **driving** .	$S[ng]\backslash NP$	PP
3	Is that **what** all this noise is about ?	$S[iq]/(S[dcl]/NP)$	NP
4	I know **what** we can do .	$S[iq]/(S[dcl]/NP)$	$S[iq]/S[dcl]$
5	**Do** n't you ?	$S[tag]/NP$	$(S[q]/NP)/NP$

(1) plays on the ambiguity of *get*, which can subcategorise for either an NP or a PP to the right. In this case the former is chosen, leading to an analysis of the modifier as an NP, as in 'Can't you get *food* to the hen?' The model is similarly at work in (2), where the predicate is a progressive verb, but the model prefers prepositional predicates, as in 'Ben White is *around*'. Linguistically, there is a case for (3), where *what* is technically a noun, but the indirect question category must be used to resolve the correct long-distance dependencies. The learner has chosen the category that ignores the NP extraction, as in 'I know *how* we can do (it).' (4) is similar. It is only slightly wrong, in that the argument should leave an NP unresolved in order to allow *what* as the object of *do*. (5) is a combination of a sentence type error and argument/adjunct confusion. It interprets the whole parse as a $S[q]/NP$, as in '*Have* n't you (any friends)?'.

6.4.3.5 Not-in-set Errors

The CI implementation returns a category set of size ≤ 10 for every target word, consisting of categories and associated probabilities, as derived according to the previous chapter. In most cases the correct category is somewhere in that category set, but in 41 cases it is missing. These are the most dangerous class of errors, because the regardless of the method used for creating a lexicon out of CI-derived categories, these types of errors will always result in erroneous lexical entries. Fortunately, it is not very common.

Figure 6.4 lists the words for which CI produces not-in-set errors (bold) in their sentential contexts. There are 41 not-in-set errors in total, involving only 25 sentences. This points to the fact that certain difficult or particularly ambiguous sentences result in multiple errors.

The last example is indicative of the whole set, a casualty of pruning. Only five categories are returned in the category set, out of a possible ten, which means that the correct category is not learned and pruned: it is never learned. In order for the correct category to be learned, the chart must have the entry $S[dcl]\backslash NP$ in cell (1,4) (*was of*

1. What **shall** we do ? said Fanny to John .
2. **Shall** we hunt for eggs **in** the barn ?
3. **I** will take you **back** .
4. How safe the little chick feels **now** !
5. **Do n't you** ?
6. **Brave** old dog !
7. **What do you think** of Ponto ?
8. The sky **is** as blue as it can be .
9. Is **that what** all **this** noise **is about** ?
10. Then we will **all** join to pay for it .
11. **Oh** ! **thank** you .
12. Sweet little birds were singing **all** around her .
13. What do you **ask** for a ticket on your train ?
14. **About** what time will you get back ?
15. At half **past** eight .
16. Do you see the children **at** play ?
17. He is **always** so polite .
18. **Poor** rat !
19. Do you see what it is **made** of ?
20. What do you **think is** in it now ?
21. How hard she must work **to** feed **them all** .
22. **Thank** you very much .
23. There ! **cried** Frank .
24. You do n't try **at** all .
25. It was of no use **to try** .

Figure 6.4: The 41 not-in-set errors (bold) occur in only 25 sentences.

no use), which it does. Then it must put $(S\backslash NP)\backslash(S\backslash NP)$ in cell (5,6), which it does, using one of the non-combinatory model rules. The next step would be to apply the unary transformation $(S\backslash NP)\backslash(S\backslash NP) \rightarrow (S[to]\backslash NP)$. However, the probability that results from this is too low to make it past the n=10 pruning threshold for that cell, and the learner misses the opportunity to learn the correct category. The solution to this problem would be to raise the pruning threshold for chart cells.

Instead of interpreting these problems as a deficiency in the learner, it is possible that we could attribute them to the corpus annotation. It would be very interesting to use the learner iteratively with annotation, alternating an automated learning step with a human correction step to achieve a grammar that is optimally learnable. A similarly logical extension would be to examine multiple sentences with the same missing item at a time, along the lines of Fouvry [2003], which is touched on in chapter 8 of this thesis.

6.4.4 Discussion

This section has introduced the McGuffey corpus as a testbed for word learning. In this initial investigation of Chart Inference as a natural-language learning mechanism, we have shown that it outperforms both the very accurate but very slow BF method, and the quick and dirty RB method. Testing on the training data has allowed us to make a detailed analysis of the types of errors and the circumstances that cause them. Analysis of the errors has shown that they are caused by vagaries of the corpus and the model, and that there are no systematic deficiencies in CI as a learning algorithm. We have shown an F-score of 76.33 for returning the single highest-probability category, and established that the correct category is more likely to be in the returned category set as the beam is increased from 5 through 10. On the strength of this analysis, we will use a beam of 10 for pruning the search space in subsequent experiments.

6.5 A Second Natural Language Experiment

The second experiment is a realistic environment for word learning: the parser is initially trained on MG1, then tested on MG2. We aim to recover as many word-category pairs as possible from the N-1 sentences of MG2 and evaluate the learned categories against the gold standard labels. We can then perform a meaningful error analysis on the results, showing how the three word learning methods compare in actual practice,

in a realistic setting. Since we are not guaranteed to have access to all the necessary word/category pairs in the seed lexicon, the precision and recall values for this second experiment should be lower than the first.

6.5.1 Corpus

To complement the MG1 seed corpus, we introduce MG2, which contains 801 sentences and is annotated in the same way as MG1. As it is constructed from the second volume of McGuffey's reader, the material is more complex and the sentences are longer.

SEED Gold standard MG1lexicon.

TEST Gold standard MG2 with CCG category labels.

6.5.2 Methods

To test the limits of the learners on truly OOL words, we again train on MG1, but test instead on MG2. This learning setting allows us to return to the N-1 condition of the previous chapter, in that the target words we are trying to learn are truly OOL, and the information we have about the surrounding words might be incomplete. The remainder of the experimental setup is identical to the previous section.

6.5.3 Results

Table 6.9 shows the category match results of the three systems on MG2[5]. Out of its 801 sentences in MG2, only 107 satisfy the N-1 condition, and of those only 32 present targets for the learners, being between 2 and 10 tokens long, containing no internal punctuation or coordination. Recall is calculated out of 32 possible targets. If the 31 unique word types from these sentences were successfully added to the lexicon, 95 N-1 sentences would be available for learning on a second pass, only 12 of which pass the length and complexity requirements.

BF is perfect at producing a category set that contains the correct tag, since the target categories for MG1 are all present in MG2. It fares worse, however, when

[5]Precision, recall, and F-score for the BF method on the In Set task is 100% by definition, since all possible categories are tried, but this is not a useful comparison

	Best Guess			In Category Set		
	P	R	F	P	R	F
BF	50.00	28.13	36.00	*	*	*
RB	16.13	15.63	15.87	29.03	28.13	28.57
CI	**61.90**	**40.63**	**49.06**	**76.19**	**50.00**	**60.38**

Table 6.9: Category match results for the three systems on the McGuffey corpus, training on MG1 and testing on MG2.

		In Category Set			Best Guess		
	count	P	R	F	P	R	F
N	11	71.43	45.45	55.55	57.14	36.36	44.44
N/N	6	75.00	50.00	60.00	75.00	50.00	60.00
NP	5	1	60.00	75.00	1	60.00	75.00
$S[dcl]\backslash NP$	2	50.00	50.00	50.00	50.00	50.00	50.00
$(S\backslash NP)\backslash(S\backslash NP)$	2	1	1	1	50.00	50.00	50.00
$(S/S)/NP$	2	0	0	0	0	0	0
$S[b]\backslash NP$	1	1	1	1	1	1	1
$S[iq]/S[dcl]$	1	1	1	1	0	0	0
S/S	1	0	0	0	0	0	0
$(S[dcl]\backslash NP)/(S[to]\backslash NP)$	1	0	0	0	0	0	0

Table 6.10: CI performance on MG2 by category type.

comparing the best guesses, yielding an F-score of only 36%. RB fares poorly on both tasks, achieving F-scores of only 16% and 29%.

CI does not perform as well as in the first experiment with complete information, but easily outperforms RB, and exceeds BF in all areas. Its precision scores are again higher than its recall. CI is clearly the best method in the *Best Guess* task, with nearly 62% precision, for an F-score of nearly 50%.

We look further into the CI results in table 6.10, where we see that CI is best at learning the most common categories, such as N and N/N, but less successful at some of the more complex ones, such as $S[iq]/S[dcl]$ and $(S[dcl]\backslash NP)/(S[to]\backslash NP)$. It is noteworthy that for several of the category types, CI performed the same on the *In Category Set* and *Best Guess* tasks. These figures will be investigated further in the error analysis in the next section.

6.5.4 Analysis

In this section we will examine the 32 individual target sentences in detail. In table 6.11, each target sentence is shown with the correct category tag for the OOL word, and the treatment of the problem by each of the three learners. For each learner, we specify whether any answer has returned (*Guessed*), whether the correct answer is somewhere in the returned category set (*Present*), and whether the correct answer is the highest-ranked returned category (*Best*). The breakdown quickly shows that RB's recall problem is due to its low threshold for returning an answer; it attempts to provide output for sentences the other two methods wisely skip.

CI correctly assigns thirteen targets in the *Best* setting. Of these, seven are *N* or *NP*, two are intransitive verbs, three are adjectives and one an adverb. These tend to be the categories on which all methods perform well.

For eleven targets the CI learner returns no answer. This is because the parser cannot find any rules to make the given categories into a cohesive tree. In most cases this is due to a known word missing a category. Sentence 5 (Suddenly she went away.) is indicative of this type of error: *away* is in the MG1 lexicon as an adverb, but there is no entry for *went* (either intransitive or subcategorising for an adverb) that allows combination. As such, the insufficient coverage of the seed lexicon is at fault.

CI succeeded at *Present* but failed at *Best* in only three instances (as opposed to BF's ten.) These are listed in table 6.12. The first sentence shows the classic *N* v. *NP* confusion. For the other two, CI chooses to return a category type that is statistically more likely, but actually a valid interpretation of the target word given the context. Given the context 'I do not know X she is so silly', the learner chooses the category $S[comp]/S[dcl]$, which would be correct if the target word had been *that*. Similarly, but further removed, 'So he ran X as fast as he could .' would justify *NP* if the target had been *Java*. Some contexts are merely ambiguous when viewed in isolation. Incorporating information about morphology and capitalisation, as well as interpolating the interpretations of one word over multiple sentences would go some way towards mitigating these ranking errors.

For five sentences for which CI returned an answer, the correct category was not present in the returned category set. This is the problem area, since we generally assume that the correct answer will be somewhere in the set. First consider the sentence 'He had a little sister about two **years** old.' The parser does not know the *X years old* construction, and tries to parse it as a flat series of adjectives, rather than a nested one.

			CI			BF		RB		
	Sentence	Gold Tag	Guessed	Present	Best	Guessed	Best	Guessed	Present	Best
1	**Tea** is over .	NP	✓	✓	✓	✓	✓	✓	✓	✓
2	James White has two **dogs** .	N	✓	✓				✓		
3	But Willie did not **stir** .	$S[b]\backslash NP$	✓	✓	✓	✓		✓	✓	
4	What is the **dark** ? asked mamma .	N						✓		
5	**Suddenly** she went away .	S/S				✓		✓		
6	Then **Susie** took the little girl into the house .	NP	✓	✓	✓	✓	✓	✓		
7	He had a little sister about two **years** old .	N	✓					✓		
8	Is not the little **fellow** a brave bird ?	N	✓	✓	✓	✓	✓	✓	✓	✓
9	But the **drones** do not work .	N						✓		
10	They kept quiet for a **short** time only .	N/N						✓		
11	At that the children gave a merry **shout** .	N						✓		
12	He is **also** found by the seaside .	$(S\backslash NP)\backslash(S\backslash NP)$	✓	✓	✓			✓		
13	It was a **jolly** ride .	N/N	✓	✓	✓	✓	✓	✓	✓	✓
14	How we both **laughed** !	$S[dcl]\backslash NP$	✓					✓		
15	**After** that we had a picnic dinner in the woods .	$(S/S)/NP$	✓					✓		
16	The sun **shone** .	$S[dcl]\backslash NP$	✓	✓	✓	✓	✓	✓	✓	
17	The grass was **cut** .	N/N	✓	✓	✓	✓		✓	✓	
18	The flowers were in **bloom** .	NP	✓	✓	✓			✓		
19	He **wished** to do as he was told .	$(S[dcl]\backslash NP)/(S[to]\backslash NP)$								
20	But he was soon very **sorry** .	N/N				✓		✓		
21	Then he set it **afloat** as he had the others .	N/N	✓			✓		✓		
22	Sometimes he would teach them a **hymn** .	N	✓	✓	✓	✓	✓	✓	✓	✓
23	**After** this the old man did not have to work .	$(S/S)/NP$						✓		
24	Do you know what a **glutton** is ?	N	✓	✓	✓	✓		✓		
25	I do not know **why** she is so silly .	$S[iq]/S[dcl]$	✓	✓		✓		✓	✓	
26	He would not eat an **acorn** too much .	N	✓	✓	✓	✓	✓	✓		
27	So he ran **off** as fast as he could .	$(S\backslash NP)\backslash(S\backslash NP)$	✓	✓		✓		✓		
28	**George** said he had not .	NP				✓	✓	✓		
29	Frank was a very **talkative** little boy .	N/N	✓	✓	✓	✓	✓	✓	✓	✓
30	All but poor **Davy** .	NP						✓		
31	Then he told her all his **trouble** .	N	✓			✓		✓		
32	How she worked that **afternoon** !	N						✓		

Table 6.11

Sentence	Gold Standard	Best Guess
James White has two **dogs** .	N	NP
I do not know **why** she is so silly .	$S[iq]/S[dcl]$	$S[comp]/S[dcl]$
So he ran **off** as fast as he could .	$(S\backslash NP)\backslash(S\backslash NP)$	NP

Table 6.12: Three cases where CI fails at *Best* but succeeds at *Present*.

'How we both **laughed** !' yields $(S\backslash NP)\backslash(S\backslash NP)$ because of complex entries for *How* that are prevalent in the dialog-heavy corpus. '**After** that we had a picnic dinner in the woods.' is returned as $(S[dcl]\backslash NP)/S[comp]$, because higher frequency uses of *that* push the correct tag past the pruning threshold. 'Then he set it **afloat** as he had the others.' poses a hard sentence to parse. It contains a high level of ambiguity and room for presuming that the seed lexicon contains the necessary items, when in fact it does not. This causes the learner to yield a spurious category: $(NP\backslash S[pt])\backslash NP$. For 'Then he told her all his **trouble**.' the most likely category is $S[dcl]\backslash NP$. The correct N does not appear in the category set because the seed lexicon does not have access to the correct ditransitive interpretation of *told*.

6.5.5 Discussion

This section has shown that CI is capable of learning new lexical items in a natural language setting. Though the number of new word/category pairs learned is low, CI has proven the most effective method for acquiring them, yielding 21 new lexical items with 62% precision. It has the greatest success recovering the most common category types N and N/N. The two types of errors that pose problems for more complex tasks are the ranking problem, where the correct category is in the category set, but not given the highest probability, and the out-of-set problem, where the parser does not have access to the correct lexical entries for the context words, and is forced to interpret arguments incorrectly or piece together a spurious OOL category out of incongruous pieces. The former can only be addressed by starting with a better model, and the latter, with a better lexicon.

6.6 The special case of *bet*

As a supplement to the evaluation above, we use one last experiment to measure a phenomenon that did not occur in either of the above experiments on the McGuffey

corpus: generation of unknown category types.

Steedman identifies *bet* as one of the few verbs with a valency of 4 [Steedman 2000]. It is OOL for the McGuffey corpus, so we run the sentences in the first column of table 6.13 through each of the learning methods and trace the process they use to decide on a solution. Given that this valency is rare, we have limited the inverse application rule to produce categories of valency 3 or lower. For this experiment we temporarily lift this restriction.

The second column of table 6.13 lists the gold-standard correct category type for *bet* in that particular context. The next three columns list the *Best Guess* returned by each of the three learners. Incorrect category types are marked with a *, and nearly correct, or plausible, categories are marked with a ?.

The BF learner only manages to return answers for half of the sentences, but gets three out of the four correct. The non-answers are produced when no available category type (from MG1) is capable of producing a well-formed parse.

In the RB condition, NP is found for every sentence. The edge selector always uses length = 3. Because the chart can find a way to parse everything after the target word into a single constituent, it does. Only for the first two sentences does it find the correct category lower in the category set, and this is because those sentences are genuinely three constituents long.

CI performs better than either of the baseline systems. For all the test sentences, the correct category is somewhere in the returned category set. In the first three cases as well as the fifth, the best guess that CI produces matches the answer. The sixth and eighth contexts (?) are correct except for some confusion between N and NP. The fourth (*) has the rightward constituents correct, but has the result wrong. This is due to the learner deciding that the whole sentence is most likely to be an NP, which is common in the training corpus. The seventh (*) has the wrong subcategorisation frame, choosing to interpret 'two cents you can' like 'two cents you gave me', using a unary rule to transform the $S[dcl]$ to $NP\backslash NP$.

6.7 Discussion

This chapter has shown in three experiments that CI exceeds the RB and BF baselines in terms of lexical coverage, and more importantly, that it does not suffer from any systematic deficiencies that prevent it from covering the full complexity of a natural English corpus. The first experiment showed CI to exceed the baseline learners in

Sentence	Correct	CI	BF	RB
I **bet** you ran .	$(S[dcl]\backslash NP)/S[dcl]$	$(S[dcl]\backslash NP)/S[dcl]$	$*(S/S)\backslash NP$	$*NP$
I **bet** that you ran .	$(S[dcl]\backslash NP)/S[comp]$	$(S[dcl]\backslash NP)/S[comp]$	$(S[dcl]\backslash NP)/S[comp]$	$*NP$
I **bet** him you ran .	$((S[dcl]\backslash NP)/S[dcl])/NP$	$((S[dcl]\backslash NP)/S[dcl])/NP$	$((S[dcl]\backslash NP)/S[dcl])/NP$	$*NP$
I **bet** him that you ran .	$((S[dcl]\backslash NP)/S[comp])/NP$	$*(NP[nb]/S[comp])/NP$	none	$*NP$
I **bet** two cents you ran .	$((S[dcl]\backslash NP)/S[dcl])/NP$	$((S[dcl]\backslash NP)/S[dcl])/NP$	$((S[dcl]\backslash NP)/S[dcl])/NP$	$*NP$
I **bet** two cents that you ran .	$((S[dcl]\backslash NP)/S[comp])/NP$	$?((S[dcl]\backslash NP)/S[comp])/N$	none	$*NP$
I **bet** him two cents you ran .	$(((S[dcl]\backslash NP)/S[dcl])/NP)/NP$	$*((S[dcl]\backslash NP)/NP)/NP$	none	$*NP$
I **bet** him two cents that you ran .	$(((S[dcl]\backslash NP)/S[comp])/NP)/NP$	$?(((S[dcl]\backslash NP)/S[comp])/N)/NP$	none	$*NP$

Table 6.13: Learned categories for all contexts of *bet*. The categories in the last three columns are the best guess given by each system.

coverage while remaining competitive on speed. It also showed that the average rank of the correct category in the returned category set is 1.4, allowing future experiments to set this parameter accordingly.

The second experiment extended the learning task to a more traditional setting, where the lexicon did not necessarily contain all the items necessary for parsing the test sentences. CI still outperformed both BF and RB, though the F-scores were lower than in the first experiment. It showed that limiting the learning setting to prioritise precision has a marked effect on recall, indicating that a much larger corpus will be needed to meaningfully measure its ability to actually improve upon an existing lexicon.

Finally, an analysis of the treatment of the high-valency word *bet* proved that CI is able to learn completely novel category types, though its ability to choose the correct category is directly influenced by the accuracy of the parsing model provided.

Chapter 7

Lexicon Recovery: CCGbank

7.1 Introduction

To test whether Chart Inference presents a tractable solution for increasing the scope of a parser with no new labelled data, we run a series of experiments using CCGbank as our seed, training, and test corpora. Because we have gold standard category labels and head-dependency information for the sentences, we can measure performance over all stages of the system. This experiment examines whether CI produces a better lexicon, and therefore a better parser, than self-training over the same data. In addition, we compare CI's performance to an oracle setting, to tease out the finer details of the impact of CI over a finite training corpus.

7.2 Related Work

In the context of parsing, self-training (ST) is the process of using a trained parser to produce new parses with which the next generation of parser can be trained. There have been mixed results for parser coverage and domain adaptation over the years: McClosky et al. [2006] and Reichart and Rappoport [2007] showed improvement, where Steedman et al. [2003a] and Charniak [1997] did not.

McClosky et al. [2006; 2008] investigated the vagaries of ST using a generative parser in combination with a discriminative reranker. Their battery of tests determine the gains from ST to be additive and attributed to a combination of circumstances: First, that ST increases the amount of evidence for additional possible dependencies between pairs of known words (biheads), and second, that it influences the reranker more than it does the parser, by giving more evidence for non-generative features.

They also concluded that more evidence in general slightly helps to reduce search errors. Overall, this investigation showed that when the seed is small, ST really helps by learning new words, but when the seed is large, biheads are more important, because the rare words are dealt with by other processes in the parser and reranker. The McClosky et al. [2006] experiments indicate that it is effective in some cases, but fails in others. They attribute this behaviour to a confluence of circumstances particular to their learning setting, which has the benefit of a discriminative re-ranker, both in the parsing case and in the learning case [McClosky et al. 2008].

Reichart and Rappoport [2007] investigate how ST contributes to domain adaptation in four experimental settings: using in-domain or out-of-domain data for the initial training and ST steps. They aim to improve a generative PCFG parser (Collins Model 2), focusing on reducing the cost of producing the labelled data needed to result in target F-scores (75-80%). Using a small seed (0-2000 annotated sentences), they show that ST can yield improvement in F-score and coverage in all four of their experimental settings. Their major claim is that using ST in the case where both data sets are in-domain means you only need to start with half the number of labelled sentences to achieve the same F-score. They also found that a single pass over the all of data is more effective than incremental processing over subsets (a claim echoed by Steedman et al. [2003b]), and that 3000 sentences of unlabelled training data is insufficient for producing a wide-coverage parser.

7.3 Corpus

For this experiment we use the CCGbank corpus [Hockenmaier 2003], which was semi-automatically translated from the Penn Treebank [Marcus et al. 1993]. CCGbank is split into 24 sections, of which §02-21 are traditionally used for training, 00 for development, and 23 for evaluation. We use CCGbank §02-21, containing 39604 sentences, as the seed corpus. The original parser is trained with these gold-standard parse trees, which have been semiautomatically generated as described in chapter 2. We report evaluation on §00 and 23, which contain 1913 and 2407 sentences, respectively.

CCGbank's semi-automatic provenance means that it contains many of the same annotation quirks and errors of the original Penn Treebank, and some further quirks introduced by the translation process. We understand that Hockenmaier attempted to keep these to a minimum, but nevertheless, there are some structures in the corpus that

Section	S	W tokens	W types	C types	$W\|C$ pairs
02-21	39604	929522	44210	1286	74669
00	1913	45422	7878	394	11282
23	2407	55371	8392	-	-

Table 7.1: Distributional analysis of the CCGbank corpus over sentences (S), words (W) and categories (C).

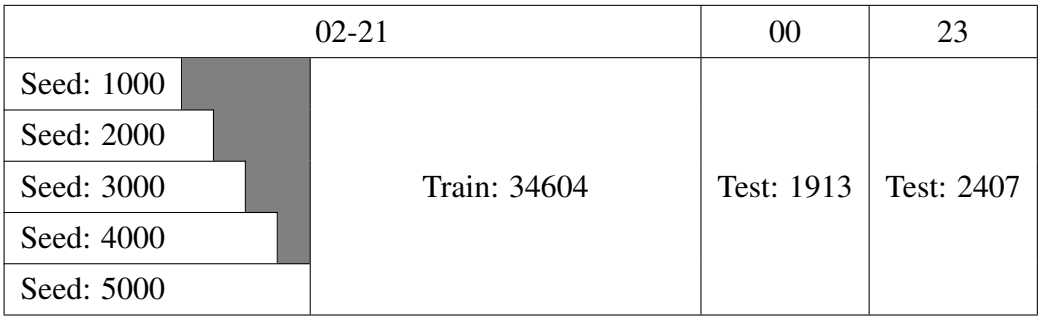


Figure 7.1: Division of the CCGbank corpus into training and test segments for the lexicon recovery experiments, with sentence counts.

are nonstandard.

Table 7.1 summarises the contents of CCGbank. We do not report the number of unique category types or lexical items for §23, because it is closed and Hockenmaier [2003] does not supply the relevant data files for it.

Figure 7.1 illustrates how the CCGbank corpus is broken down into Seed, Train, and Test sections. We run five settings to compare how the seed size affects the quality of learned sentences, and the resultant quality of the parsed test set. The Train and Test sets remain the same throughout the experiment. This formulation is non-standard, but designed to test the effect of changing the number of seed sentences on the quality of the learning step. At each Seed setting, both the parser and the lexicon are trained on only the seed section of CCGbank, and the remainder of the corpus is left closed for use in the training set. A more standard experiment may randomly select N random sentences to serve as the Seed set, and the rest as the Training set, but this allows us to make direct comparisons over consistent subsets of data.

SEED Five seeds from 1000-5000 sentences from the beginning of CCGbank §02-21.

TRAIN 34604 sentences from the end of CCGbank §02-21.

TEST CCGbank §00 and 23.

As the seed size increases from 1000 to 5000 sentences, so does the number of learning opportunities. We expect the F-score over dependencies on test sets 00 and 23 to be greatest in the 5000 condition.

7.4 Methods

Five separate experiments are run in isolation, each starting with a seed size between 1000 and 5000 sentences. First, a parser is trained on only those sentences of the seed. In the learning step we use CI to obtain a weighted list of possible lexical entries for the target word, then add those into the working lexicon and return the highest-probability parse from the StatOpenCCG parser. We then add all these new parsed sentences to the training set and, at the final evaluation step, we retrain StatCCG with the expanded set of sentences and parse §00 and §23 using only the original parsing mechanism. These final parses are evaluated against the gold-standard dependency information provided by Hockenmaier and Steedman [2007]. The ST baseline is calculated in the same way, running the seed-trained parser rather than CI over the 34604 training sentences and integrating the new parses to retrain before the final test phase.

One major issue to contend with when adding to an existing lexicon and model is how to incorporate the new items, which are likely of inferior quality, with the known items, which in our case are not quite gold-standard, but at least high-quality. In this case we generate new parsed sentences and then concatenate them with the training data, as in ST, but there is a great deal of further investigation to be done on how best to extend a lexicon and model on the fly, which would enable much faster, online learning. It may be possible to incorporate more sophisticated weighting in the statistical model, regarding lexical items gleaned from labelled and unlabelled sentences [Nigam et al. 1999; Deoskar 2008; Deoskar et al. 2011]. We leave this as future work, and discuss further ramifications of this issue in chapter 9.

7.4.1 Parameter Settings

The timeout is set to 200000ms, and the category set beam to 10, returning an answer set of up to ten categories for each target word to the parser. The StatCCG word frequency threshold is set to 5 for all phases of the experiment, so a token must be seen at least 5 times for its categories to be included in the lexical model, as opposed to the POS model.

7.4.2 Baselines

We compare CI to self-training by substituting a normal parsing run for the CI step, and include three other benchmarks: the parser trained on seed lexicon with no learning (lower bound), the parser trained on all Seed+34604 sentences (upper bound) and the Oracle, which is constructed by keeping a list of all the sentences CI attempts, and putting their gold-standard trees into retraining, rather than the CI-produced trees. The Oracle figures represent an upper bound given CI's low recall. They are what we could expect if CI were perfect at producing parse trees, and the difference between Oracle and CI gives some indication of the quality of the parses CI produces.

We anticipate the ST baseline will be high because the training and test corpora share a domain, following Reichart and Rappoport [2007], but the interesting area is between ST, which is cheap to run, and the Upper Bound, which is impossible to achieve without labelled data. The experiment examines whether CI is an effective way of improving upon ST without the use of any additional resources.

7.4.3 Evaluation

We evaluate against the gold-standard dependency annotation of CCGbank, both the development set §00 and the closed test set §23 [Clark and Curran 2003; Rimell and Clark 2008b; Rimell et al. 2009]. Unlabelled and Labelled dependencies are available for both sections, from which we can calculate precision, recall, and F-score.

7.5 Results

The results of this experiment show that CI is more effective than ST in improving the performance of a baseline parser. Furthermore, increasing the number of sentences used to train the seed lexicon from 1000 to 5000 yields real improvement in F-score, which improves with the size of the seed.

Table 7.2 reports precision, recall and F-score over labelled and unlabelled dependencies for §00. All three metrics strictly improve as the seed size is increased, as we would expect. Unlabelled scores are consistently higher than labelled, due to the increased complexity of the labelled task.

The upward trend is echoed in table 7.3, where we can see the seed lexicon getting bigger in terms of both tokens ($|L_{to}|$) and types ($|L_{ty}|$), and the opportunities for learning in the N-1 setting increasing as a result. The number of new training sentences

Seed	UP	UR	UF	LP	LR	LF
1000	83.37	71.80	77.15	72.33	62.29	66.94
2000	84.00	75.09	79.29	73.58	65.78	69.46
3000	85.27	76.94	80.89	75.50	68.12	71.62
4000	85.70	79.14	82.29	76.40	70.55	73.36
5000	86.14	81.37	83.69	76.89	72.63	74.70

Table 7.2: Labelled and Unlabelled Precision, Recall, and F-score of CI over dependencies as a function of seed size, for test section §00.

Seed	$\|L_{ty}\|$	$\|L_{to}\|$	N-1 sents	#CI	#ST
1000	893	7406	3657	812	32448
2000	1635	12089	5875	1311	32940
3000	2185	15537	7462	1516	33390
4000	2643	18672	8271	1494	33606
5000	3137	21560	8935	1523	33717

Table 7.3: Corpus statistics as a function of seed size, including number of types $|L_{ty}|$ and tokens $|L_{to}|$ in the seed lexicon, number of N-1 learning opportunities out of a possible 34604, and training sentences parsed by CI and ST.

	UF	LF	UF	LF
Seed (Baseline)	83.45	74.62	83.63	75.36
ST	82.26	72.94	82.66	74.39
CI	83.69	74.70	83.66	75.25
Oracle	83.74	74.93	83.87	75.53
Upper Bound	89.38	82.32	89.01	82.38

Table 7.4: Labelled and Unlabelled F-score over dependencies in §00 and 23, starting with 5000 seed sentences.

added by CI and ST are markedly different, although they do increase with the seed size. ST yields twenty to forty times more new sentences than CI: the former is able to handle most of the sentences in the training set, while the latter is limited to only those N-1 sentences for which CI produces a parse.

Table 7.4 shows labelled and unlabelled F-score for the setting that performed best. ST effects a decrease in parsing performance with respect to the baseline setting using just the 5000 sentences to train the seed lexicon. CI, however, shows a numerical improvement on both metrics over §00 and UF over §23, though the difference is not statistically significant. LF on §23 is lower than the baseline, but not so low as ST. The CI F-scores are approaching the Oracle, but are still some way off from the Upper Bound, showing that there is still more to be learned by increasing the recall, so as to include more sentences in the retraining phase.

Figure 7.2 tracks the change in labelled F-score over §00 as the size of the seed is increased. When the initial lexicon is small, CI and ST degrade performance against the Seed. As the lexicon is made stronger, the gap between Seed and CI narrows, so that CI just outperforms the Seed at 5000 sentences. In contrast, the gap between Seed and ST stays relatively constant. A similar relationship is shown for §23 in FIgure 7.3, though all the curves are closer together. The one difference, however, is that ST outperforms CI for low amounts of seed sentences, but the relationship reverses for the larger seeds.

7.6 Analysis

The central result of this chapter is CI's relationship to the Oracle, since this indicates how well the learner is performing in terms of its parse accuracy, a proxy for error rate.

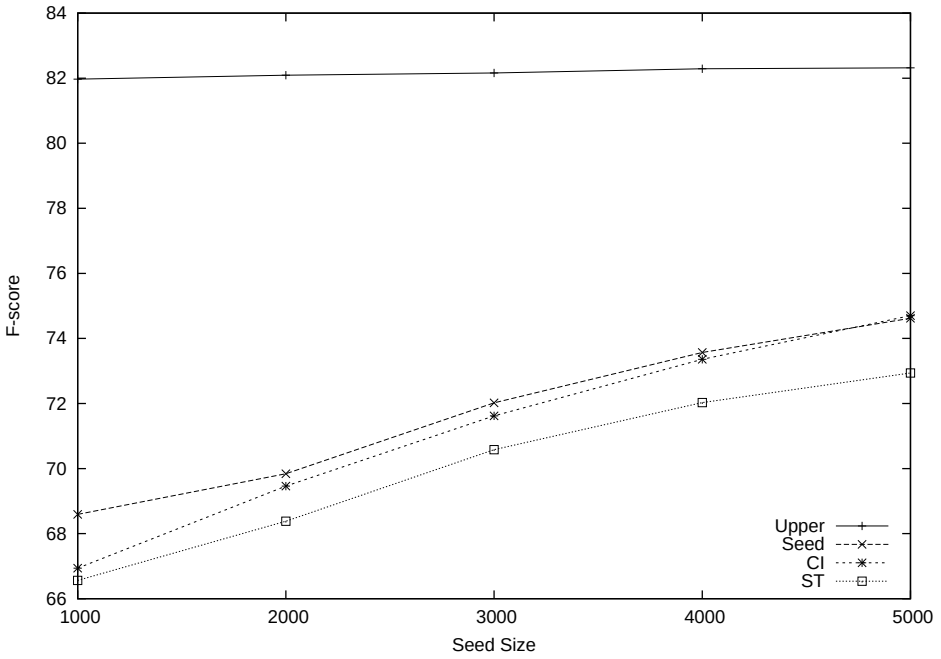


Figure 7.2: Impact of CI on Labelled F-score, depending on seed size, §00.

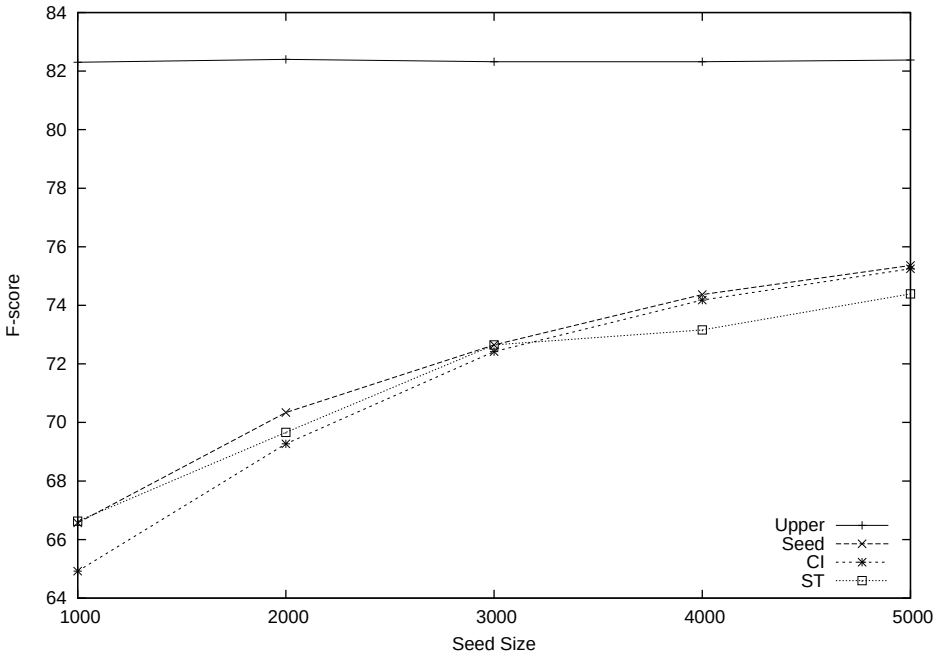


Figure 7.3: Impact of CI on Labelled F-score, depending on seed size, §23.

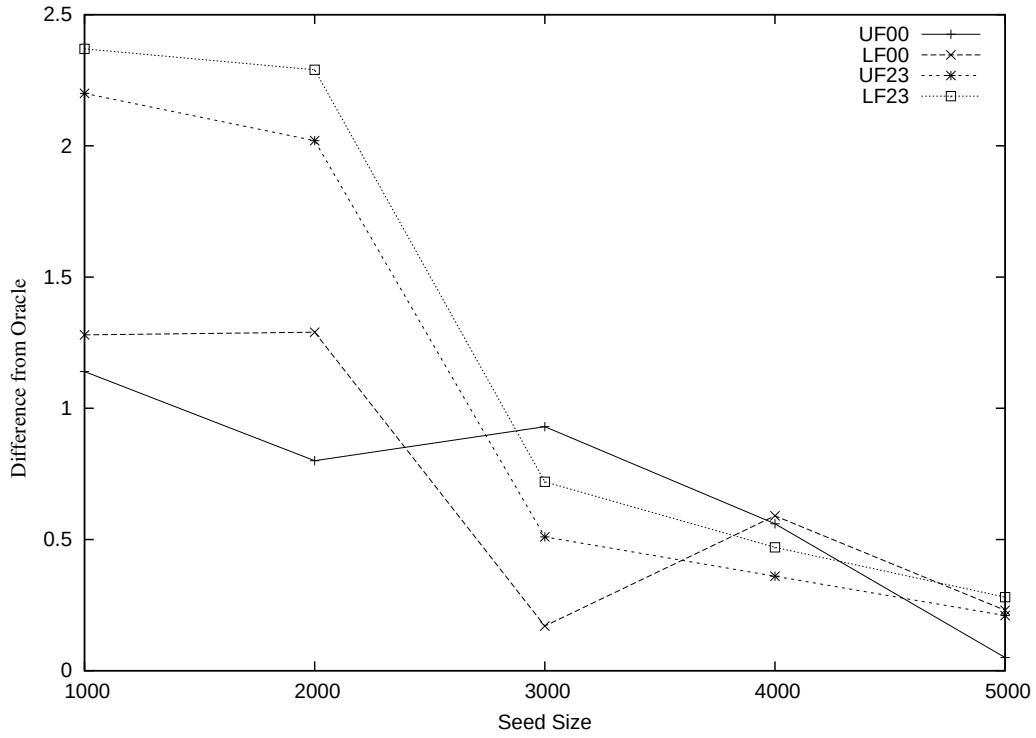


Figure 7.4: Difference in labelled and unlabelled F-score between CI and Oracle as seed size increases, over dependencies in both §00 and 23.

The Upper Bound is very high, since it assumes that all of the training sentences can be handled by the learner, while the Oracle shows that restricting the number of learning opportunities greatly mitigates the potential effect of CI on the resultant lexicon. Remember that a large number of sentences are rejected by CI because they contain more than one unknown word, or are unparseable due to the large search space.

The CI and Oracle results are very close, showing that this implementation of CI is near 100% precision when it comes to parsing unseen sentences. As the number of seed sentences increases, the gap between CI and Oracle decreases, showing that the parsing accuracy is improved with a larger seed.

Figure 7.4 shows the difference between CI and Oracle for Labelled and Unlabelled F-score over both §00 and §23. Note that the values on the y-axis represent the difference between the CI and Oracle values, rather than real F-score. While the extent of all four gaps decrease as more seed sentences are made available, §23 UF and LF do so more consistently. §00 begins with a smaller gap than §23, but does not decrease

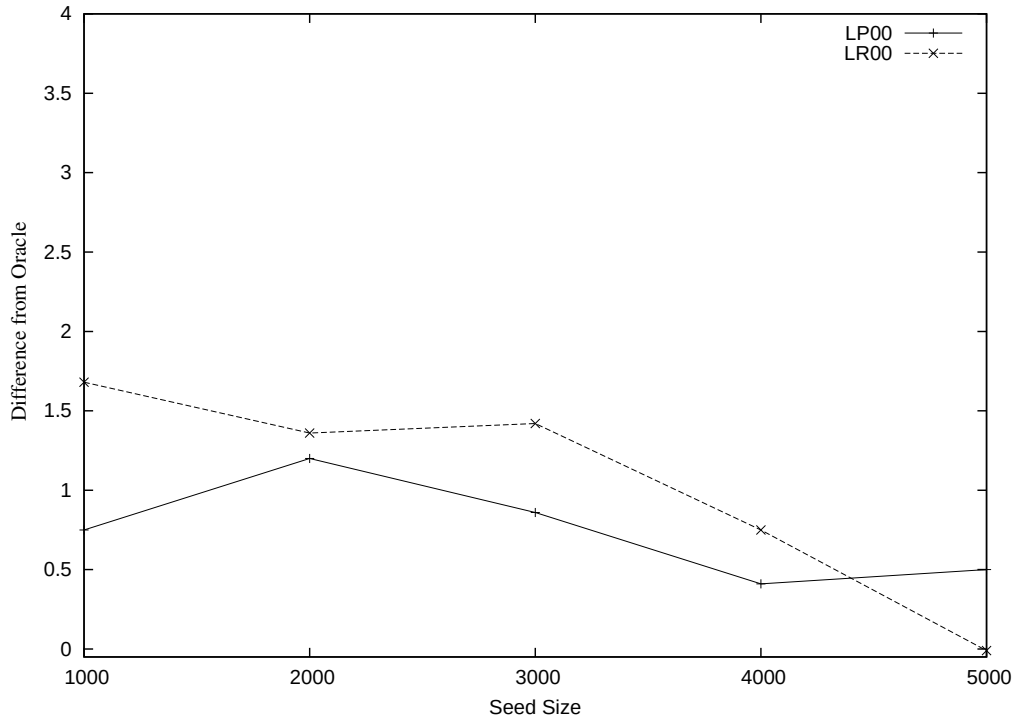


Figure 7.5: Difference in labelled precision and recall between CI and Oracle over dependencies in §00.

as smoothly: UF experiences a rise from 2000 to 3000, and LF from 1000 to 2000 and 3000 to 4000. This shows that increasing the number of seed sentences has more of an effect on §23, while §00 is affected more erratically.

Figures 7.5 and 7.6 examine this behaviour in finer detail, comparing the gap in LP and LR between CI and Oracle as the seed size is increased. The recall curves in the two figures are very similar, while the precision seems to be the cause of the difference.

7.7 Discussion

In this chapter we have compared Chart Inference (CI) to self-training (ST) in the task of recovering as much lexical information as possible from the unlabelled sentences of CCGbank. Our results show that the self-trained parser performs worse than the baseline in all cases. While ST yields more retraining sentences than CI, as reflected in the last two columns of table 7.3, the latter extends the coverage of the baseline parser without affecting its precision to the extent that ST does. This highlights the problem of bad data. When the seed corpus is large, any new parses added take some of the probability away from the lexical pairs we consider gold. Because the parser itself is

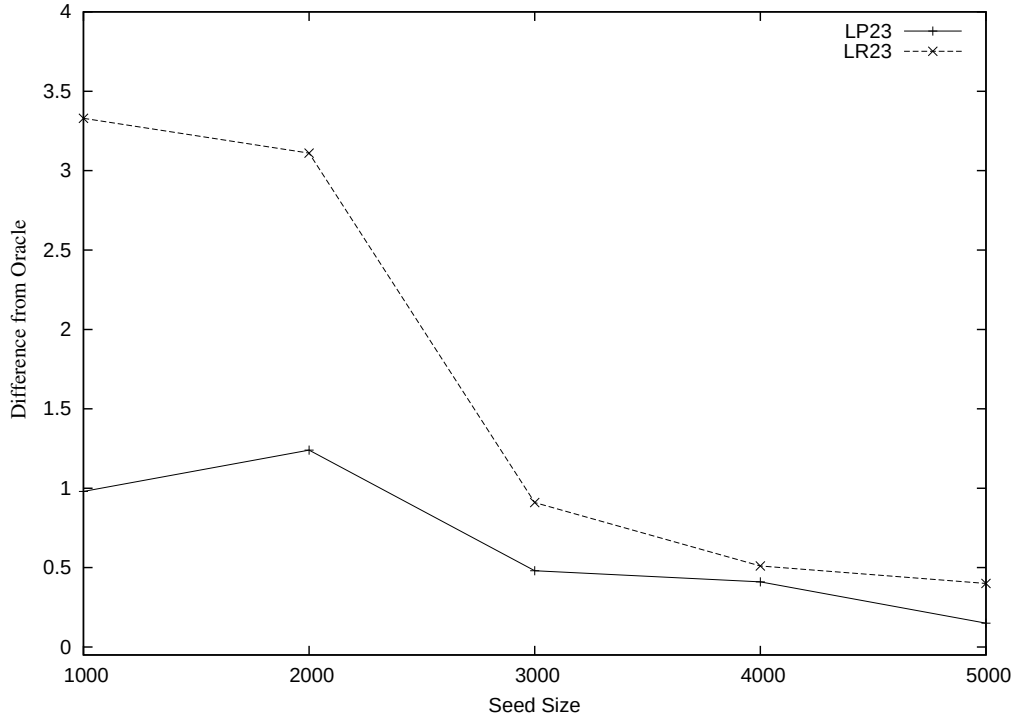


Figure 7.6: Difference in labelled precision and recall between CI and Oracle over dependencies in §23.

not very good, we cannot trust what we learn from applying it.

There are two ways of explaining this behaviour. The first blames the sheer number of additional parses for the downfall of ST. Because CI simply returns fewer sentences, it disrupts the probabilities in the seed parser least. The second is that CI is only learning from the most dependable of sentences, as determined empirically in the preceding chapter. ST is free to make more mistakes per sentence. It is probably a combination of these effects that leads to the performance differences we see in table 7.4.

In addition to showing that CI is better than plain ST, we consider it a valuable result to have confirmed the intuition that smaller seeds are less reliable, both as a basis for ST and for CI. In the next experiment, we examine how CI fares when the baseline parser is very good.

Chapter 8

Domain Adaptation: The TREC QA Corpus

8.1 Introduction

The task of domain adaptation for parsing questions is both interesting and useful, and Chart Inference could be one tool in a strategy for solving it. In this chapter we attempt to recover the lexical items in the TREC QA corpus that are missing from CCGbank. The seed lexicon is large and contains all the category types necessary for parsing most declarative sentences, but question words are rare and have not been attested with all the necessary category types for a Question Answering corpus.

The structure of this task boils down to the discovery of only a small number of OOL word/category pairs, and the learner has access to a relatively large number of short sentences that contain the target words. It is also a contemporary real-world learning task that is interesting outwith the field of parsing.

8.2 Related Work

Clark et al. [2004] identified the problem with using news data to train a parser for a question answering task as the lack of lexical support for question words. Some lexical types were missing entirely. The lexicon for CCGbank §02-21 contains 12 WH-question types, notably lacking some important ones. Clark et al. note the absence of one category in particular: $(S[wq]/(S[dcl]\backslash NP))/N$, the category needed for *What* in the sentence *What President became Chief Justice after his presidency?*

They attempt to adapt the discriminative C&C parser [Clark and Curran 2007] to

the QA domain by retraining on 500 hand-labelled question sentences, then automatically parsing and hand-correcting an additional 671. The entire set was then used in conjunction with CCGbank §02-21 to train a final parsing model. Their per-word accuracy rose from a 68.5% baseline to 94.6% for the newly trained model.

In this experiment, we examine how close we can get to those results by using CI to learn WH-question words from the unlabelled question corpus. If successful, this would eliminate the human-annotation step for domain adaptation.

8.3 Corpora

We trained the initial parser on the full CCGbank training set, consisting of 39603 sentences[1]. It is important to note that this training corpus, even though it is considered to be of very high quality, contains only 93 questions in total, so it is not surprising that it is missing some whole category types for question words. It also reinforces the fact that this is a domain-adaptation task.

The learning corpus contains 1328 questions from the TREC training set. We use the same test set as Rimell and Clark [2008b], which consists of 488 unique question sentences, each starting with *What*, *When*, *How*, *Who* or *Where*. Table 8.1 sets out the distribution of question types in the test corpus. The third column marks whether the necessary category type is represented in the Seed lexicon.

SEED StatCCG lexicon and parsing model trained on CCGbank §02-21, with frequency threshold 5.

TRAIN 1328 questions from the TREC corpus, unlabelled.

TEST 488 unique held-out questions.

8.4 Methods

For this experiment, we trained the parser on CCGbank §02-21 with a word frequency threshold of 5[2]. It produces partial parse charts in the cases where all words in the sentence are in-lexicon, except for the WH-word target, for which the learner attempts

[1] In order to train the full model, we needed to remove sentence 7713 from the CCGbank training set. Somehow this sentence was making the rule creator in StatOpenCCG crash, but the issue is believed to have been addressed in subsequent versions of the parser.

[2] StatCCG requires a parameter to trade off between training the lexicon and the POS-backoff.

	Count	In S?	Category	Example
What	231	+	$S[wq]/(S[q]/NP)$	What is the Keystone State ?
	61	-	$(S[wq]/(S[q]/NP))/N$	What continent is Scotland in ?
	51	-	$(S[wq]/(S[dcl]\backslash NP))/N$	What gift is proper for a 1st anniversary ?
	11	-	$(S[wq]/S[q])/N$	What year did Nintendo 64 come out ?
	2	+	$S[wq]/(S[dcl]\backslash NP)$	What lays blue eggs ?
When	29	+	$S[wq]/S[q]$	When did Einstein die ?
	12	-	$S[wq]/(S[q]/NP)$	When was the Hellenistic age ?
How	9	-	$(S[wq]/(S[q]/(S[adj]\backslash NP)))/(S[adj]\backslash NP)$	How fast is the speed of light ?
	9	-	$((S[wq]/(S[q]/NP))/N)/(NP/N)$	How many hearts does an octopus have ?
	5	+	$(S[wq]/S[q])/(S[adj]\backslash NP)$	How fast is the world spinning ?
	5	+	$S[wq]/S[q]$	How did Eva Peron die ?
	2	-	$((S[wq]/PP)/((S[q]/PP)/(S[adj]\backslash NP)))/(S[adj]\backslash NP)$	How far is a nautical mile ?
	2	-	$(((S[wq]/PP)/((S[q]/PP)/NP))/N)/(NP/N)$	How many liters are in a gallon ?
	1	-	$(S[wq]/(S[dcl]\backslash NP))/NP$	How much of the ozone layer is depleted ?
Who	16	+	$S[wq]/(S[dcl]\backslash NP)$	Who discovered quarks ?
	15	+	$S[wq]/(S[q]/NP)$	Who was Galileo ?
Where	10	+	$S[wq]/S[q]$	Where did the U.S. Civil War begin ?
	9	+	$S[wq]/(S[q]/NP)$	Where is Big Ben ?
Which	5	+	$(S[wq]/(S[dcl]\backslash NP))/N$	Which river runs through Dublin ?
	2	+	$(S[wq]/(S[q]/NP))/N$	Which country 's major export is coffee?
	1	-	$S[wq]/(S[q]/NP)$	Which was the first shuttle ?

Table 8.1: Distribution of Wh-question category types in the test corpus.

to return a category motivated by that context. We run the learner on the TREC Question-Answering corpus [Rimell and Clark 2008b], aiming to learn from the set of 149 sentences that contain the word/category pair *What*:=$(S[wq]/(S[dcl]\backslash NP))/N$. The WH-questions are annotated with CCG categories, but not full trees, and have no dependency information. We therefore only evaluate the output with regard to individual categories matching the gold standard.

Only two out of the five categories needed to parse *What*-questions are paired with the word *What* in the CCGbank seed lexicon:

Object question category $S[wq]/(S[q]/NP)$ – *What is a verb?*

Subject question category $S[wq]/(S[dcl]\backslash NP)$ – *What has 18 legs and catches flies?*

The main focus of this experiment is on the missing subject WH-element extraction (SWHE) category $(S[wq]/(S[dcl]\backslash NP))/N$, of which there are 51 in the test set. This particular category was chosen as a point of investigation because it is OOL in CCGbank yet is common enough to meaningfully evaluate.

Subject WH-element extraction category (SWHE) $(S[wq]/(S[dcl]\backslash NP))/N$ – *What kind of man eats the last cupcake?*

8.4.1 Parameter Settings

Because we know that every sentence in the learning corpus is a Wh-question, we change the parameters slightly. We replace the end-punctuation distribution derived from the training corpus with a single value: $P(S[wq]|?) = 1$. This is an artificial environment, but the change means that the top ten returned categories are completely different. We also change the rules file. Since the CCGbank-trained rules file contains 200 rules, we reduce these to a more manageable set of just the 8 combinators, plus unary rules for $N \rightarrow NP$ and $PP \rightarrow adjuncts$. The version currently running uses the 27-rule mid set. We keep the unknown rule backoff at 0.00001, since this is comparable to expected values in the baseline StatCCG model.

The parser is initialised by training on the 39,604 sentences of CCGbank §02-21. It produces partial parse charts in the cases where all words in the sentence are in-lexicon, except for the WH-word target, for which the learner attempts to return a category motivated by that context. Since the target WH-words are technically in-lexicon, we force the learner to treat them as unknown by replacing them with a dummy variable in the training corpus.

		Baseline			Self-trained			
		P	R	F	P	R	F	ΔF
Who	(31)	48.15	41.94	44.83	48.15	41.94	44.83	0
What	(356)	57.02	54.78	55.87	68.91	66.01	67.43	11.56
When	(41)	58.54	58.54	58.54	46.34	46.34	46.34	-12.20
Where	(19)	68.42	68.42	68.42	68.42	68.42	68.42	0
Which	(8)	71.43	62.50	66.67	71.43	62.50	66.67	0
How	(33)	16.13	15.15	15.63	19.35	18.19	18.75	3.12
Total	(488)	54.60	52.25	53.40	62.45	59.63	61.01	7.61

Table 8.2: Category Match performance on question test corpus by baseline parser, trained on CCGbank §02-21 with word frequency threshold 30, and the same parser self-trained on the QA corpus, for WH-words only. Difference in F-score is shown in the final column.

8.4.2 Baselines

We compare the CI-retrained parser to two baselines: the original performance by the CCGbank-trained parser alone (the seed) and self-training (ST), as described in chapter 3. We follow the recommendation of McClosky et al. [2008] that the best performance is achieved when all the training sentences are parsed at once, rather than incrementally. The domain-adaptation work of Reichart and Rappoport [2007] previously discussed is also directly relevant, as they showed ST to be effective in the case where the original training data was out-of-domain with respect to the test set, and the ST data in-domain.

Table 8.2 gives a detailed description of the baseline performance, and what can be achieved through ST on the 1328 TREC questions. We show category match accuracy over the question words only, with baseline F-scores ranging from 15 to almost 70%, depending on the word. In total, the baseline §02-21 StatCCG parser achieves just over 50% for precision, recall, and F-score over the 488 question words of the corpus. The low recall numbers are down to the parser not being able to handle many of the sentences due to an insufficient lexicon. This problem is magnified when the POS-backoff mechanism is disabled, causing recall to fall to only 13.11%.

When we use ST to improve upon the baseline parser, the change in F-score (reported in the final column of table 8.2) is telling. For three of the question words there is no effect, but we see improvement for *What* and *How*. However, the performance

on *When* suffers almost as much, resulting in a total change in F-score of only 7.61%, which is nonetheless significant. The finer-grained results tell a clearer story, which is that ST is both powerful and destructive. It is only the relative volume of the question words that result in a positive effect overall.

8.4.3 Evaluation

We evaluate the success of CI in bootstrapping Wh-question categories from the out-of-domain corpus in two ways. First, we compare the CI output to the gold standard categories labelled in Rimell and Clark [2008a]. Second, we add the parsed questions into the training set, then retrain and finally retest the parser.

We can report precision, recall, and F-score over category matches in the test set. Dependency information is not included in the CCG annotation of the TREC corpus. A fine-grained analysis is also available, where matches over a subset of categories are reported, separating the performance on all words in the test sentence from that on the set of all WH-question words, any particular WH-question word in turn, or a specific target word/category pair.

8.5 Results

CI produces a parse for 26 of the 149 questions. It only gets the correct what-category for four of these.

Table 8.3 shows the change in F-score throughout this experiment. BL is the baseline condition, where the accuracy is predictably high over all the words in the sentence, but lower when we examine the question words only. It is most telling that the baseline F-score over words that should be tagged with the SWHE category $((S[wq]/(S[dcl]\backslash NP))/N)$ is extremely low. In fact, that seven percent represents only a handful of instances of *Which*, and none of *What*. Applying CI to the problem results in statistically significant increases in all metrics, but the biggest gain is in the last. When we first apply CI, then self-train over the full training corpus, we further increase all metrics, and again the largest gain is over the target category type specifically, as shown in the last column of table 8.3[3].

[3]We also ran the experiment using ST only, which performed better than CI alone, but only over a different set consisting entirely of seen categories. We do not report those figures here because they are not commensurable with the CI results.

	All Words	POS=WHQ	Word=What	Cat=SWHE
BL	84.31	53.40	55.87	7.84
ST	**87.47**	**61.01**	**67.43**	7.84
CI	86.59	56.19	60.83	52.94
CI+ST	87.03	59.54	65.42	**58.82**

Table 8.3: F-score over category matches in the QA test corpus for the baseline parser, ST, CI, and a combination of CI+ST. The last three columns each represent a different subset of word types. The most improvement is shown in the last, which considers only the subject WH-element extraction categories.

The results in table 8.3 can be more clearly interpreted when viewed in conjunction with table 8.4, which shows the differences in the impact on the lexicon between baseline (BL), Chart Inference (CI), and the combined method of CI and ST (CI+ST)[4]. CI leaves the initial distribution unchanged while adding seven more category types. One of these is the category we are interested in: $(S[wq]/(S[dcl]\backslash NP))/N$, which is previously associated with *Which* in the baseline lexicon. The other six are spurious categories, but they are invented categories and have low counts. Combining the learning mechanisms by running first CI, and then ST, has the effect of introducing the category we need, and then elevating the counts. The probability for $S[wq]$ is elevated as well, as a result of misparses, but the whole process results in better category matches over the test set that we saw in table 8.3.

It is now clear that the ST condition results in probabilities for the *What*-categories to be elevated. This directly effects performance when parsing *What*-questions, since now these categories will be preferred by the generative model over other non-wq categories that do not apply to the QA test corpus. This explains why overall performance increases in the ST condition, but performance on our specific SWHE category does not.

When we use the methods in combination, we not only get the benefit from CI of adding previously unknown categories, we also get a probability boost from ST in two of the categories. This adjusted model can now clearly outperform CI in all metrics, and ST in the SWHE case. The remainder of ST's advantage we can only attribute to luck, since the improved probabilities are the result of incorrect parses. We must assume in this case that it is the sheer amount of exposure to WH-questions that causes

[4] *What* has 31 categories in total in the baseline lexicon; here we show only the [wq] types.

		$P(W\|C)$			F			$P(C\|W)$		
?	C	BL	CI	CI+ST	BL	CI	CI+ST	BL	CI	CI+ST
R	$S[wq]$	0.09	0.09	0.17	1	1	2	0.006	0.005	0.002
R	$S[wq]/PP$	0.6	0.6	0.6	3	3	3	0.019	0.016	0.003
*	$(S[wq]/PP)/N$	1	1	1	1	1	4	0.006	0.005	0.004
*	$S[wq]/(S[adj]\backslash NP)$	0.5	0.5	0.5	1	1	1	0.006	0.005	0.001
	$S[wq]/(S[dcl]\backslash NP)$	0.37	0.37	0.86	22	22	239	0.137	0.118	0.225
*	$S[wq]/(S[dcl]/NP)$	1	1	1	1	1	8	0.006	0.005	0.008
*	$(S[wq]/(S[dcl]/NP))/N$	0.5	0.5	0.5	1	1	1	0.006	0.005	0.001
*	$S[wq]/(S[ng]\backslash NP)$	1	1	1	1	1	2	0.006	0.005	0.002
R	$S[wq]/S[poss]$	0.83	0.83	0.83	5	5	5	0.031	0.027	0.005
*	$S[wq]/S[q]$	0.03	0.03	0.12	2	2	9	0.012	0.011	0.008
	$S[wq]/(S[q]/NP)$	0.64	0.64	0.97	16	16	331	0.099	0.086	0.312
	$(S[wq]/(S[q]/NP))/N$	0.36	0.36	0.95	4	4	136	0.025	0.021	0.128
	$(S[wq]/(S[dcl]\backslash NP))/N$	-	**0.5**	**0.96**	-	**4**	**75**	-	**0.021**	**0.071**
*	$S[wq]/N$	-	1	1	-	8	12	-	0.043	0.011
*	$(S[wq]/S[dcl])/N$	-	1	1	-	8	28	-	0.043	0.026
*	$(S[wq]/N)/N$	-	1	1	-	4	7	-	0.021	0.007
?	$(S[wq]/S[inv])/N$	-	1	1	-	3	78	-	0.016	0.074
*	$(S[wq]/(S[dcl]/(S[pt]\backslash NP)))/N$	-	1	1	-	1	2	-	0.005	0.002
*	$(S[wq]/((S[dcl]\backslash NP[expl])/NP))/N$	-	1	1	-	1	12	-	0.005	0.011

Table 8.4: Lexical category distribution for the word *What* in the baseline §02-21 of CCGbank (BL), after Chart Inference (CI), and after first applying Chart Inference, then self-training (CI+ST). Column 1 classifies low-frequency categories as rare (R), spurious (*) or plausible (?). Categories above the middle line are present in the Baseline lexicon; below are induced.

ST to succeed. This is confirmed by the lack of similar model differences in the CI+ST case: since the learner does not have the opportunity to make these bad (but useful) decisions, it cannot benefit from them.

This shows us that it's not enough just to change the lexicon of the StatCCG parser. We have to change the model parameters as well, and since that is just like retraining, and we have a full parse of the sentence already, we just concatenate the newly parsed sentences with the initial seed corpus and retrain. It is also relevant to note that three of the category types are justifiable, though rare, uses of the word *What*. These are marked (R) in table 8.4. It means that six of the *What*-categories in the Baseline lexicon are in effect correct. Of the seven categories added by CI, one is the target SWHE category, and one $((S[wq]/S[inv])/N)$ is a plausible interpretation of the same function within the bounds of the baseline CCGbank lexicon. It will be discussed in further detail in the test sentence analysis.

8.6 Analysis

We separately analyse the parses produced by CI at the training phase and those produced by the retrained parser at the test phase.

8.6.1 Training Parses

Of the 149 subject WH-element extraction questions, 30 contain enough known lexical items to afford learning opportunities. Only four of these correctly assign the SWHE category to the target *What*. Here we examine the errors incurred by the learner.

The 26 "incorrect" sentences use one of six categories for *What*. Table 8.5 shows the distribution of category errors and an example of each. First, eight questions lead to a category very similar to our SWHE target, but missing the extracted *NP*: $(S[wq]/S[dcl])/N$. In every case, this is due to the rightmost constituent being resolved as a complete declarative sentence. In three instances the incorrect lexical item *is*:=$S[dcl]/(S[pss]\backslash NP)$ is to blame, once *has*:=$S[dcl]/(S[pt]\backslash NP)$, and twice *'s*:=$NP[nb]/N$.

The lexical item *'s*:=$NP[nb]/N$ is linguistically unmotivated and could therefore be considered a bad lexical entry. It arises from two distinct sentences in CCGbank: *yesterday 's edition* and *next year 's Lower House Elections*. In both cases the correct category should have been $(NP[nb]/N)\backslash NP$, but these two errors contribute

Count	Correct C	Returned C	Example
8	$(S[wq]/(S[dcl])\backslash NP))/N$	$(S[wq]/S[dcl])/N$	What city ['s airport is named Logan International]:=$S[dcl]$?
8	$(S[wq]/(S[dcl])\backslash NP))/N$	$S[wq]/N$	What [automobile company [makes]:=N the Spider]:N ?
4	$(S[wq]/(S[dcl])\backslash NP))/N$	$(S[wq]/N)/N$	What American landmark [stands]:=N on Liberty Island ?
3	$(S[wq]/(S[dcl])\backslash NP))/N$	$(S[wq]/S[inv])/N$	What English word [has the most letters]:=$S[inv]$?
1	$(S[wq]/(S[dcl])\backslash NP))/N$	$(S[wq]/(S[dcl]/(S[pt]\backslash NP)))/N$	What metal [has]:=$(S[dcl]/(S[pt]\backslash NP))/NP$ the highest melting point ?
1	$(S[wq]/(S[dcl])\backslash NP))/N$	$(S[wq]/((S[dcl]\backslash NP[expl])/NP))/N$	What country [is]:=$((S[dcl]\backslash NP[expl])/NP)/(S[adj]\backslash NP)$ home to Sabena Airlines ?
1	$(S[wq]/(S[dcl])\backslash NP))/N$	$(S[wq]/PP)/N$	What percent [of the U.S. is African American]:=PP ?
26			

Table 8.5: Training stage errors.

overmuch to the lexicon, causing too tempting a path for the learner, since the determiner category is so probable in most situations. Likewise, *is*:=$S[dcl]/(S[pss]\backslash NP)$ and *has*:=$S[dcl]/(S[pt]\backslash NP)$ are motivated by a single sentence each in CCGbank: ... *250 times as much as* ***is*** *contained in* ... and ... *longer than* ***has*** *been the case* ... respectively.

The next eight errors miss the declarative sentence constituent completely, choosing instead to parse it as a relative clause or other modifier for the extracted N. These errors caused by the verb of the sentence being interpreted as an adjective five times (*have, produced, made, designed, created*), and as a noun once (*makes*). This ambiguity is central to linguistics and difficult to rectify. For example, *made*:=N/N is in CCGbank four times: either in the phrase *domestically made N* or *newly made N*. Here it is the annotation scheme that is deficient. Precisely the same motivation leads the learner down the wrong path to $(S[wq]/N)/N$ three times.

The next most common error is linguistically motivated. Rather than interpret the embedded sentence as declarative, the parser uses *has*:=$S[inv]/NP$ three times to interpret it instead as an inverted sentence. In essence, it cannot see the difference between *What companies have them?* and *What choice have they?* when the *NP*s differ in case, but lack the distinction within the existing CCGbank structure. A relatively simple extension to the annotation of the treebank would obviate this problem. Two of the remaining errors are due to rare categories for *is* and *has* being preferred over the more common correct ones, and behave similarly to the $S[inv]/NP$ condition.

One sentence was labelled with $(S[wq]/PP)/N$, in an example that clearly shows the downside to using a generative model. The parser has preferred rare categories for common words: *the*:=$(S[adj]\backslash NP)/N$ and *of*:=$PP/(S[adj]\backslash NP)$, causing the whole sentence to be parsed consistently, and with high generative probability, though incorrect. Fortunately, the EM parse trainer is smart enough not to include this entry into the lexicon, since it is rare and inconsistent with the rest of the model.

These errors show us how sensitive the learner is to mistakes in the seed lexicon. A more useful experiment would include using a seed lexicon derived from CCGbank more strictly.[5] A smaller and more accurate lexicon would reduce the incidence and

[5] As is, StatCCG trains its parser using a frequency threshold over word types, but not word/category types. This has the detrimental effect of allowing mistakes to intrude unduly on the model, but moving to a word:category threshold would increase the number known words with missing categories, which are much harder to recover from. In essence, it is easier to recover from seeing too much than seeing too little.

C	BL	CI+ST
$(S[wq]/(S[dcl]\backslash NP))/N$	0	26
$*S/S$	7	1
$*(S/S)/NP$	1	0
$*S[wq]$	2	0
$*(S[wq]/PP)/N$	2	0
$*S[wq]/(S[adj]\backslash NP)$	5	0
$*(S[wq]/(S[dcl]/NP))/N$	1	0
$*S[wq]/(S[dcl]/NP)$	11	1
$*S[wq]/(S[dcl]\backslash NP)$	9	0
$*(S[wq]/(S[q]/NP))/N$	5	0
$*(S[wq]/S[dcl])/N$	0	9
$?(S[wq]/S[inv])/N$	0	3
$*(S[wq]/((S[dcl]\backslash NP[expl])/NP))/N$	0	2
No category returned	8	9

Table 8.6: Fine-grained category attribution showing how baseline sentences are re-analysed by the CI+ST parser.

severity of these mistakes. Additionally, we could try to make the learner more robust in the face of these errors, but that would take further engineering, and the mistakes that first caused the problem would still be in the lexicon to cause problems in later stages of the parsing pipeline.

8.6.2 Test Parses

Of the 51 test questions that use *What*:=$(S[wq]/(S[dcl]\backslash NP))/N$, the retrained parser returns a parse for 42 of them, 26 of which are correct. Table 8.6 shows the difference in errors between the baseline and best case CI+ST over the 51 extraction test sentences. In addition to making fewer errors, the CI+ST condition makes *better* errors: the last three category types in table 8.6 have the N constituent correct, which means that the parses maintain at least one correct dependency relation, even if they do not recover the extraction of the sentential object.

Of the previously known categories, the ST step overwhelmingly prefers three categories: one subject extraction category $S[wq]/(S[dcl]\backslash NP)$ and two object extraction $S[wq]/(S[q]/NP)$ and $(S[wq]/(S[q]/NP))/N$. The remaining categories have been pre-

viously classified in table 8.4 as either rare (R), spurious (*), or plausible (?). Rare categories, like $S[wq]$ are used for specialised cases (the sentence *What?*) which occur in PTB, but not in the QA corpus. Spurious categories, like $(S[wq]/PP)/N$ exist in the baseline parser, arising from errors in either the original PTB, or the translation to CCGbank. $S[wq]/S[q]$ is only used where $S[wq]/(S[q]/NP)$ is meant, but fails to capture the extraction. $S[wq]/(S[dcl]/NP)$ is a misinterpretation of sentences requiring $(S[wq]/(S[dcl]\backslash NP))/N$, but without capturing the extracted N.

Five spurious categories are also introduced by the CI learning step. $(S[wq]/S[dcl])/N$ and $(S[wq]/((S[dcl]\backslash NP[expl])/NP))/N$ are spurious forms of $(S[wq]/(S[dcl]\backslash NP))/N$ that arise when the constituent directly right of the target is misparsed; the former misses the extraction and the latter adds an extra dummy subject. $S[wq]/N$ occurs when the main verb of the sentence is treated as a participle, forming a complex nominal argument. $(S[wq]/N)/N$ and $(S[wq]/(S[dcl]/(S[pt]\backslash NP)))/N$ are caused by similar verbal ambiguity.

The classification of $(S[wq]/S[inv])/N$ as a plausible category is linguistically motivated. Rather than interpret the embedded sentence as declarative, the parser uses *has*:=$S[inv]/NP$ to interpret it instead as an inverted sentence, because of CCGbank's case insensitivity as previously discussed. As such, it duplicates the work of the target $(S[wq]/(S[dcl]\backslash NP))/N$, because the constituents $S[dcl]\backslash NP$ and $S[inv]$ are often synonymous in practice, and result in the same dependency structure.

As seen in table 8.4, the distinction between rare and spurious categories cannot be made on frequency alone, but the best categories are the ones with the highest frequency. Plausible categories can be considered spurious for the sake of parsing, but are linguistically interesting, and if they are frequent enough, that is possibly an indication that the structure of the lexicon or the grammar is non-optimal.

8.7 Discussion

This chapter has assessed Chart Inference in a domain adaptation task, attempting to recover lexical items that are known to be missing from the baseline lexicon. There are two goals of this task: achieving the correct category inventory, and calculating reasonable model probabilities over that inventory. CI aids in the former and ST in the latter. When a system uses both, we see combined effects.

Analysis has shown that CI has mixed success with constructing category types for question words. This is undoubtedly due to the parser being trained on CCGbank,

which contains only 93 question sentences. Because the model trained from this data is overwhelmingly prepared for non-question structures, it is not surprising that decisions it makes in unfamiliar genres are often erroneous.

However, we have shown that CI can overcome the deficiencies in both the lexicon and the model to learn new category types for question words. When used in conjunction with ST, CI presents a valuable framework for domain adaptation in the case where whole category types are missing from the lexicon. Equipped to invent category types based on surrounding context, CI is capable of finding the necessary lexical items, but it is encumbered with restrictions on the number and complexity of training sentences it is capable of learning from. ST remedies this by propping up the learning process with volume. Sheer exposure, whether with correct parses or not, does in this case aid in adapting the baseline lexicon to a question corpus.

By using CI in combination with ST over a small, specialised corpus, we have achieved three points improvement in F-score over category matches in the TREC QA test corpus, from a baseline of 84.31% to 87.03% in the combined condition. This is motivated by a dramatic increase in the performance over one rare category type, from 7.84% in the baseline to 58.82%. CI introduces new lexical entries for the known word *What*, and then ST uses the expanded lexicon over the same corpus to adjust the model probabilities, yielding a ten-point improvement in F-score for that single word.

The fact that CI is capable of constructing new category types from context makes it a potentially valuable tool in tackling the long tail of the lexicon. It is important in particular for CCG, since the category inventory is so large and varied that the number of unique unseen category types is likely to be large as well. The next chapter makes a start in improving the coverage of the StatCCG parser by automatically discovering OOL lexical items.

Chapter 9

Wide-coverage Lexicon Extension: Gigaword

9.1 Introduction

Up to this point, the thesis has explored the elements of lexicon learning both in general, and within the scope of the CI algorithm, varying the corpus size, composition of the seed lexicon, and learning parameters. However, the true test of the system is on large-scale data. There are several major obstacles to this task, as we have shown in the previous chapters: First, the frequency threshold must be set to allow the optimum number of correct entries into the lexicon. Second, if the seed lexicon is very large, it is difficult to measure any appreciable impact on parse quality, which is why we also evaluate on lexicon quality. That leads into the third obstacle, which is that the recall of CI is very low, necessitating a very large set of learning data to have any impact on precision at all.

This chapter investigates the efficacy of CI as a method f or extending the coverage of the existing StatCCG lexicon trained on CCGbank by producing additional training parses from the unlabelled English Gigaword corpus [Graff and Cieri 2003]. In this experiment we target the specific set of words in §00 that are OOL in §02-21, ignoring numbers. We mine Gigaword for learning opportunities for these words in particular, to estimate how much unlabelled text we would need to look at to cover all word-category pairs in §00.

Given that §00 and 23 are the standard evaluation corpora, we expect success to look like a very small change in parsing behaviour over these sets, accompanied by an increase in the size of the lexicon. When we examine the new word/category pairs that

have been added by the learning process, we should see mostly correct entries, where the incorrect entries have very low probability as compared to other words of the same category.

9.2 Corpora

The corpora for this experiment represent the state-of-the-art in labelled and unlabelled data. CCGbank §02-21 is used for the original training set, which dictates the seed lexicon. We also use §00 and 23 as an in-domain test set, which is made possible by the presence of gold-standard lexical labels and full parse trees. Gigaword is the source of raw training sentences. However, the speed of the implementation limits us to using just a portion of the whole corpus at present. Table refgigsizes compares the size and complexity of the resources used in this experiment: although the number of sentences differs widely, the average complexity of the sentences are roughly comparable.

For running on the Gigaword corpus we use the same restrictions as indicated by the previous experiments: sentences of length between 3 and 10, containing no internal punctuation or coordination [, ; : and] as established in chapter 6. In addition, we restrict the learning environment further to adapt to the NYT structure by ignoring any short sentence (<5 words) beginning with the word *By*. This is because these "sentences" are most often bylines, and learning the names from them would take a disproportionate amount of processing time and most likely result in incorrect categories being learned, because the result is not *S*. When we restrict the sentence length these bylines are overrepresented in the learning data.

The input sentences are annotated with the POS tags provided by Gigaword. They are not used by the CI algorithm, but are incorporated into the output parse trees to be compatible with the original StatCCG training corpus.

The baseline parser is trained on CCGbank §02-21, with a frequency threshold of 5. The list of missing words is generated by examining §00 for any word type not listed in the baseline lexicon. There are 2409 such word types, representing 3321 distinct tokens. Missing items occur 1.38 times on average. There are 45422 total tokens in §00 of CCGbank, 11282 word-category pairs, and 7878 individual word types. Only 65 different category types are represented in the Missing set. The most common missing categories are listed in table 9.2, dominated by N and N/N, which constitute 41% and 36% of the types, respectively. This is expected, as they are the most populous open classes. The other 23% cover the remaining 63 category types, including many verbal

	CCGbank §02-21 (Seed)	Gigaword NYT §10-14 (Train)	CCGbank §00 + 23 (Test)
Sentences	39604	10,677,224	4320
Words/Sentence	23.47	22.41	23.33

Table 9.1: Size and complexity comparison of the corpora in the Gigaword experiments.

Rank	Freq	C	Rank	Freq	C
1	1040	N	11	18	$(S[dcl]\backslash NP)/PP$
2	915	N/N	12	17	$(S[adj]\backslash NP)/PP$
3	58	$(S[dcl]\backslash NP)/NP$	12	17	$(N/N)/(N/N)$
4	55	$S[adj]\backslash NP$	13	16	$(NP\backslash NP)/(NP\backslash NP)$
5	52	$(S[ng]\backslash NP)/NP$	14	14	$(S\backslash NP)\backslash(S\backslash NP)$
5	52	$(S[b]\backslash NP)/NP$	14	14	$S[b]\backslash NP$
7	43	$S[pss]\backslash NP$	16	12	$(S\backslash NP)/(S\backslash NP)$
8	33	$NP\backslash NP$	16	12	$((S[ng]\backslash NP)/PP)/NP$
9	22	$S[dcl]\backslash NP$	18	11	$(S[ng]\backslash NP)/PP$
10	19	$S[ng]\backslash NP$	20	10	$(S[pss]\backslash NP)/PP$

Table 9.2: Top 20 most frequent category types from w/c pairs in §00 that are missing from §02-21.

types, for which CCG makes a finer-grained distinction than the other open classes.

The training corpus we use for this experiment is a subset of Gigaword: §10-14 of the New York Times corpus, which have been automatically POS-tagged, and contain around two million sentences each, for a total of nearly 10 million sentences. The word and sentence breakdown of the five sections is outlined in figure 9.3. In addition to restricting the number of sentences shown to the learner, CI also rejects the majority of the sentences because they do not contain enough in-lexicon words.

The test corpus consists of §00 and 23 of CCGbank, for which we have gold standard dependency and category data. We conduct the error analysis on §00 only, and leave §23 closed to represent coverage on unseen data.

SEED CCGbank §02-21.

TRAIN Gigaword §NYT 10-14.

TEST CCGbank §00 and 23.

Section	Sentences	Words
nyt10	2,117,736	47,882,337
nyt11	2,094,226	47,905,885
nyt12	2,139,542	47,860,591
nyt13	2,153,149	47,847,010
nyt14	2,172,571	47,827,489
total	10,677,224	239,323,312

Table 9.3: Gigaword corpus statistics.

9.3 Methods

The baseline parser is trained on §02-21, as in the previous experiment. Then the learner is run over a subset of Gigaword, only attending to sentences that afford learning opportunities for the tokens in the list of the 2409 OOL types. We target only these types in order to drive development towards a measurable improvement over §00, given that the learner is slow. Of course, we expect the system to learn other correct category types for these tokens, as well as influencing the model at retraining time to introduce previously discounted word/category pairs present in the baseline corpus.

The threshold problem from chapter 8 still stands: for a word to become accepted into the retrained lexicon, it must have been seen five times. Some of the words in the Missing list have been seen in §02-21, but not enough times to reach the threshold. For some words, a single additional instance will be enough to tip it from the POS set into the lexical training set. Others would have to be seen up to five times in the CI output before they are learned in the retraining phase. Considering the number of times each OOL word occurs in §02-21 (a third are never seen at all), the minimum number of CI-output sentences to get all the OOL words above the threshold and into the lexicon is 9390.

Even when we know what words to look for, the task of lexicon extension is still a very difficult one. It is unreasonable to expect, even if all the unlabelled sentences are parsed perfectly, that we would see a significant difference in the final parse results, given StatCCG's double hurdles of the word-frequency threshold and the EM lexicon trainer. With so well-trained a baseline lexicon, the reasonable thing to look for is an increase in the size of the lexicon, coupled with no significant decrease in the parse results over the evaluation corpora. Coverage over the long tail is hard to detect when measuring over a small corpus. Therefore we rely on an error analysis of the results as

the main contribution of this chapter.

9.3.1 Parameter Settings

The word frequency threshold is set to 5, which is not the optimal configuration for StatCCG[1], but allows for a good balance of missing words and a low bar to lexicon change. Sentences must be between three and ten tokens long, containing no commas or conjunctions.

The learner produces a ranked category set of up to 5 for each target word. These are added into the hypothesis lexicon with their probabilities and a full parse is produced. The category that is ultimately chosen is the one that produces the highest-probability parse, not necessarily the one ranked the highest in the learner's set. This can be seen as a reranking step. The learner is prohibited from using the end-punctuation category type (.). The only sentences considered are the ones containing exactly one word from the OOL list, while the rest of the words have at least one lexical entry in the StatCCG lexicon with a word frequency threshold of 5.

We only allow the learner to attempt sentences for which exactly one word is OOL. The learning parameters further limit the usable sentences by length, stopwords and target words. Runtime issues such as memory capacity, timeouts, and parse failures also cause many sentences to be rejected. After taking all of these restrictions into account, CI only produces parses for around 1000 sentences of our 10 million-setentence section of Gigaword, which will be discussed in the learner results in §9.6.1.

9.3.2 Baselines

As a baseline for this experiment, we use the existing POS backoff system internal to StatCCG. This is already a very high score to try to beat, but it is accompanied by a lexical error analysis. We set the word frequency threshold to 5 in the baseline to provide a fair comparison for the trained conditions. We also have the gold-standard dependency and category information for §00 and 23 for comparison, though this is not considered a baseline. For lexical analysis, we compare learned lexical items to a baseline strategy of labelling all new words *N* or *NP*.

It is important to note here the difference between POS-backoff and self-training as learning strategies, since they are conflated in several phases of our experiments.

[1] Hockenmaier's best advice calls for the threshold set at 30.

POS-backoff is an online strategy for resolving the roles of OOL items at parse time, while both CI and ST enable the parser to feed decisions about OOL items back into the training corpus, thereby influencing future decisions. Whenever we use the original parser to produce new training material, it is essentially self-training, but we make the distinction between doing so while employing only the built-in backoff mechanisms of the original parser (ST) and intervening with our own category-finding algorithm (CI).

9.3.3 Evaluation

Several forms of evaluation are necessary to assess the behaviour of CI across multiple phases of learning. First we evaluate on whether the correct categories were learned for the target words, both as tokens in the individual training sentences, and as types in the retrained lexicon. The gold standard is the given category type in §00. In cases where a word occurs twice, we accept either of the category types given, and we also count these as two separate lexical items for the learner to recover.

Second, we examine the lexicon that results from retraining on the learned sentences. Each new entry is evaluated by hand for accuracy, as well as participation in the final parse output.

Third, we evaluate the retrained parser on §00, measuring the change in head dependency recovery and category matches, over the set of all sentences, the set of sentences containing OOL words, and finally the set of OOL tokens itself.

9.4 The Lexicon

First we report results and analysis of the lexicon resulting from CI learning. This can be thought of as running the training phase of the experiment, then stopping to inspect the learned lexical items before proceeding to use them to parse the test set. In practice, we run the whole experiment first, then inspect each phase, so as not to unfairly manipulate the results. Evaluation of the learned lexicon provides insight into the linguistic soundness and internal consistency of the learning system.

9.4.1 Lexicon Results

The size of the lexicon has risen from 38054 in the baseline case to 38197 after retraining on the learned sentences, resulting in a net change of 143. 121 lexical entries and

	All Sents	-Q
$N\|NP$	51/99 (51.51%)	34/52 (65.38%)
CI	103/121 (85.12%)	**82/86 (95.35%)**

Table 9.4: Manually evaluated lexical accuracy of CI vs. $N|NP$ baseline, i.e. whether or not the learned lexical items are valid, without reference to the contexts that motivated them. Results are reported for the full output (All Sents) and the condition where sentences ending in a question mark were removed from the training set (-Q).

24 POS entries have resulted from the CI learning step, and two POS entries have been obsoleted by learning: PDT:=$(S/S)/(S/S)$ and VBD:=$S[adj]\backslash NP$.

The 121 new lexical entries were examined by hand for plausibility and it was determined that 103 were valid and 18 were spurious. The learning step has increased the size of the lexicon by 0.4%, with 85% accuracy. We compare this to the $N|NP$ baseline in table 9.4. The $N|NP$ baseline is calculated by assuming any lexical entry found by CI as correct if either if it is assigned either N or NP. We count each word type only once, irrespective of how many different category types CI has learned for it, so the original 121 entries are reduced to 99. We see that the accuracy for lexical entries rises from 51.51% for the baseline to 85.12% for CI. This shows that, even though N and NP are the most common categories for an OOL item, CI is a much better method for word learning.

We also report figures for a condition resulting from the insights of the error analysis. The -Q column shows the results for the same experiment when sentences ending in a question mark are filtered out of the training corpus. This reduces the number of lexical entries to 86, and the number of unique word types to 52. This causes the lexical accuracy of $N|NP$ to rise to 65.38%, since the reduced set has a higher concentration of nominal types, but at the same time the lexical accuracy CI -Q achieves 95.35%.

The evaluation of the lexical items produced by retraining the parser over the CI-produced sentences was performed manually. First, the 24 new POS entries are listed in table 9.5. Of the 24 new POS entries, only four were deemed reasonable. Most were obviously spurious. POS information is ignored by CI, as can be seen here: nominal and adjectival POS types have been given verbal categories, and POS subtypes (VBG) are not limited to their appropriate categorial types ([ng]). In future work, POS information should be taken into account.

*JJ: PP/NP	*NNP: $(N/N)\backslash S[wq]$	*NN: S
*JJ: $S[dcl]\backslash S[dcl]$	NNP: $NP/(NP\backslash NP)$	*NN: $S[dcl]\backslash NP$
*JJ: $S[to]\backslash NP$	NNP: $NP[thr]$	*NN: $S[dcl]\backslash S[dcl]$
*VB: PP/NP	*NNP: $(S[b]\backslash NP)/N$	*NN: $S[dcl]\backslash S[q]$
VB: $S[dcl]\backslash NP$	*NNP: $S[dcl]\backslash NP$	*NN: $S[ng]$
*VB: $S[dcl]\backslash S[to]$	*NNP: $S[dcl]\backslash S[wq]$	NNS: NP
*VBG: $S[dcl]/N$		*NNS: S
*VBG: $S[wq]/PP$		*NNS: $(S[dcl]/NP)/S[inv]$
*VBZ: $(S[dcl]\backslash S[wq])/N$		*NNS: $S[pt]\backslash NP$

Table 9.5: The 24 new POS entries introduced by CI over Gigaword.

Add:= $(S[b]\backslash NP)/NP$	converts:= N
adventure:= N	converts:= $((S[dcl]\backslash NP)/PP)/NP$
Ames:= N	converts:= $(((S[dcl]\backslash NP)/PP)/PP)/NP$
Ames:= N/N	*counterattack:= $S[dcl]\backslash S[q]$
batter:= N	cream:= N
batter:= $(S[b]\backslash NP)/NP$	cream:= N/N
battling:= N/N	curse:= N
battling:= $(S[ng]\backslash NP)/NP$	Directed:= N/N
battling:= $(S[ng]\backslash NP)/PP$	dish:= N
blind:= N/N	dish:= $(S[b]\backslash NP)/NP$
blind:= $S[adj]\backslash NP$	*entertain:= $(S[dcl]\backslash S[dcl])/NP$
Broader:= N/N	entries:= N
Brunswick := N/N	*entries:= $NP\backslash NP$
*Brunswick:= $(N/N)/(N/N)$	Fifteen:= NP (N)
Carroll:= N	flashes:= N
Carroll:= N/N	flashes:= $S[dcl]\backslash NP$
Cemetery:= N	flashes:= $(S[dcl]\backslash NP)/NP$
Cemetery:= N/N	Got:= $S[pss]\backslash NP$
cheaply:= $(S\backslash NP)\backslash(S\backslash NP)$	Got:= $(S[pt]\backslash NP)/NP$
cheese:= N	*Hampton:= $S[dcl]\backslash NP$
cheese:= N/N	Harper:= N
...	

Table 9.6: The 121 new lexical entries introduced by CI over Gigaword.

chocolate (N):= N/N
*Clemens:= $(S[dcl]\backslash NP)/PP$
clowns:= N
*combinations:= $(N/S[inv])\backslash S[wq]$
companions:= N
composer:= N
contemplate:= $(S[b]\backslash NP)/NP$
contemplate:= $(S[b]\backslash NP)/(S[ng]\backslash NP)$
contemplate:= $(S[dcl]\backslash NP)/NP$
convenient:= $S[adj]\backslash NP$
convenient:= $(S[adj]\backslash NP)/PP$
Less:= $(S[adj]\backslash NP)/(S[adj]\backslash NP)$
lively:= N/N
Meet:= $(S[b]\backslash NP)/NP$
Meet:= $(S[dcl]\backslash NP)/NP$
*Mercer:= $S[dcl]\backslash S[q]$
mixture:= N
musician:= N
Music:= N
Music:= N/N
Newsweek:= NP
*Newsweek:= $(S[dcl]\backslash S[wq])/N$
nuts:= N
*obsessed:= $(S[dcl]\backslash S[wq])/PP$
offense:= N
Palestinians:= N
pastry:= N/N
ponder:= $(S[b]\backslash NP)/NP$
ponder:= $(S[dcl]\backslash NP)/(S[ng]\backslash NP)$
Portland:= N
*Portland:= $(N/N)/(N/N)$
Portland:= $NP\backslash NP$
Harper := N/N
hid:= $(S[dcl]\backslash NP)/PP$ (/NP)
Houghton := N (N/N)
ideology:= N
Illustrated:= N
Illustrated:= $N\backslash N$
Israelis:= N
Jenkins:= N
Kaplan:= N
Kids:= N
Less:= N/N
quitting:= $S[ng]\backslash NP$
refrain:= N
refrain:= $S[b]\backslash NP$
refrain:= $(S[b]\backslash NP)/PP$
reminds:= $((S[dcl]\backslash NP)/PP)/NP$
reminds:= $((S[dcl]\backslash NP)/S[dcl])/NP$
rookie:= N
rookie:= N/N
*SAT:= $(S[dcl]\backslash S[dcl])/NP$
sauce:= N
Schlesinger:= N
*Send:= NP
sings:= $S[dcl]\backslash NP$
sings:= $(S[dcl]\backslash NP)/NP$
Song:= N
*Song:= $NP\backslash NP$
Talk:= N
Talk:= N/N
*Test := $S[adj]\backslash NP$
THAT:= NP $((NP\backslash NP)/(S[dcl]\backslash NP))$
Top:= N/N
...

Table 9.6: The 121 new lexical entries introduced by CI over Gigaword.

prosecuting:= $(S[ng]\backslash NP)/NP$	*Top:= $S[adj]/S[adj]$
Q.:= N/N	Wallace:= N
qualified:= N/N	wandering:= $S[ng]\backslash NP$
qualified:= $S[adj]\backslash NP$	*Winning:= $NP[nb]/N$
qualified:= $(S[dcl]\backslash NP)/PP$	*Winning:= $S[wq]/PP$
Quite:= N/N	Yukon:= N
Quite:= NP/NP	Yukon:= N/N
Quite:= $(S/S)/(S/S)$	

Table 9.6: The 121 new lexical entries introduced by CI over Gigaword, continued.

Table 9.6 lists the new lexical entries, marking the incorrect ones with a *. One common pattern within the learned lexical items is that nominal and adjectival forms are often paired. This is quite a frequent behavioural alternation, but is more generalisable with a unary rule $N \rightarrow N/N$. It is related to the fact that these two categories are also the most frequent in the set. The most impressive result is that multiple verbal patterns have been learned for verb forms, such as *ponder* and *reminds*. Even Noun-Verb ambiguity has come through, as in *refrain*, which is learned as the nominal type NP, and two verbal types $S[b]\backslash NP$, and $(S[b]\backslash NP)/PP$.

Table 9.7 lists all of the lexical entries from the §00 OOL set that the CI retraining process adds to the lexicon. In the broad case, 30 of the lexical entries pertain to words that are OOL in §00. Nine of these are exact matches for the gold standard w/c pairs in §00, giving a match rate of 30%, and a further seven are valid w/c pairs, so the overall lexical accuracy over the target set is 53%.

When the question filter is applied, only 10 of the target words are found, but at a 50% success rate for matching the target category as well. A further four are valid categories for OOL words, though not a match to their gold standard senses. Removing questions from the learning set reduces the yield by two thirds, but increases the success rate from 53% to 90%.

The remaining 93 learned lexical items result from other words in the sentence. This can be observed in such sentences as 'Swan Song for **Hampton**?' where *Hampton* is OOL, but *Song* appears in the new lexicon as well. *Song* occurs four times in §02-21: 3 times as N and once as $NP\backslash NP$. This one new instance tips the word type over the frequency threshold and causes both of those categories to appear as lexical entries for *Song*.

W	§00 Gold C	Learned C	-Q C
blind	N/N	N/N	N/N
		$?S[adj]\backslash NP$	$?S[adj]\backslash NP$
Brunswick	N	$?N/N$	$?N/N$
		$*(N/N)/(N/N)$	$*(N/N)/(N/N)$
chocolate	N/N	N/N	N/N
Clemens	N	$*(S[dcl]\backslash NP)/PP$	-
combinations	N	$*(N/S[inv])\backslash S[wq]$	-
convenient	$S[adj]\backslash NP$	$S[adj]\backslash NP$	$S[adj]\backslash NP$
		$?(S[adj]\backslash NP)/PP$	$?(S[adj]\backslash NP)/PP$
counterattack	N	$*S[dcl]\backslash S[q]$	-
curse	N	N	-
entertain	$(S[b]\backslash NP)/NP$	$*(S[dcl]\backslash S[dcl])/NP$	-
Fifteen	N	$?NP$	-
Hampton	N	$*S[dcl]\backslash NP$	-
hid	$(S[dcl]\backslash NP)/NP$	$(S[dcl]\backslash NP)/PP$	-
Houghton	N/N	$?N$	$?N$
Mercer	N/N	$*S[dcl]\backslash S[q]$	-
Newsweek	N	$?NP$	-
	N/N	$*(S[dcl]\backslash S[wq])/N$	-
obsessed	$(S[adj]\backslash NP)/PP$	$*(S[dcl]\backslash S[wq])/PP$	-
offense	N	N	-
prosecuting	$(S[ng]\backslash NP)/NP$	$(S[ng]\backslash NP)/NP$	$(S[ng]\backslash NP)/NP$
quitting	$S[ng]\backslash NP$	$S[ng]\backslash NP$	-
SAT	N	$*(S[dcl]\backslash S[dcl])/NP$	-
sauce	N	N	N
Send	$(S[b]\backslash NP)/NP$	$*NP$	-
Test	N	$*S[adj]\backslash NP$	-
THAT	$(NP\backslash NP)/(S[dcl]\backslash NP)$	$?NP$	-
Winning	$(S[ng]\backslash NP)/NP$	$*NP[nb]/N$	-
		$*S[wq]/PP$	-
match		9	5
*		14	1
?		7	4
total		30	10

Table 9.7: Comparison of OOL items learned by CI against the gold-standard labels from §00.

Total	Correct	Incorrect	C
38	38	0	N
23	23	0	N/N
6	5	1	$S[dcl]\backslash NP$
6	6	0	$(S[b]\backslash NP)/NP$
5	4	1	NP
4	4	0	$(S[dcl]\backslash NP)/NP$
4	3	1	$S[adj]\backslash NP$
3	1	2	$S[dcl]\backslash S[q]$
3	2	1	$(S[dcl]\backslash NP)/PP$
3	1	2	$NP\backslash NP$
2	1	1	$S[wq]/PP$
2	2	0	$(S[ng]\backslash NP)/NP$
2	2	0	$S[ng]\backslash NP$
2	1	1	$(S[dcl]\backslash S[wq])/N$
2	0	2	$(S[dcl]\backslash S[dcl])/NP$
2	2	0	$S[dcl]\backslash S[dcl]$
2	2	0	$((S[dcl]\backslash NP)/PP)/NP$
2	2	0	S
2	2	0	PP/NP
2	0	2	$(N/N)/(N/N)$
...	...	...	...

Table 9.8: Distribution of category types from the lexical items learned by CI over Gigaword; table shows only types that occur more than once.

In order to be included in the lexicon as a lexical entry (as opposed to POS entries) the word-category pair must reach the frequency threshold of 5. These 5+ are constituted partially by gold-standard sentences from the original CCGbank training data, and partially by new labelled data gleaned from the learning process.

9.4.2 Lexicon Analysis

We now undertake a manual analysis of the origins of all the erroneous lexical items introduced to the lexicon by the CI learning and retraining process. There are three major sources of error: mistakes in the gold-standard annotation of the training corpus, incorrect resolution of the target word as the head of the sentence, and lexical ambiguity of common structural words. Several cases suffer from more than one of these causes.

9.4.2.1 Gold standard errors

	Sentence	Gold Standard	Best Guess
1	the New **Brunswick** , N.J. company	$(N/N)/(N/N)$	$(N/N)/(N/N)$
2	The books 1,500 **entries** include ...	$NP\backslash NP$	$NP\backslash NP$
3	The **Portland** , Ore. , thrift said...	$(N/N)/(N/N)$	$(N/N)/(N/N)$
4	The 1981 novel Wedding **Song** ...	$NP\backslash NP$	$NP\backslash NP$

The first four errors are attributable to vagaries of the gold standard annotation for §02-21. In the chart above, the Gold Standard category is identical to the Best Guess category. These arise because of the manner in which the frequency cutoff is implemented. As was discussed in previous chapters, a word type must occur N times to justify lexical entries; under this number the data points count toward the POS entries. So when a word type occurs four times in the gold standard corpus, and CI introduces a fifth, the lexicon now considers lexical entries for all the categories that word type has been seen with. At the same time, those four word-category pairs are removed from the POS data. Errors 1-4 are indirectly caused by CI learning, but are ultimately the fault of the annotated data. (This phenomenon is exaggerated when we work with low word-frequency threshold values.)

9.4.2.2 Gap realised as head

	Sentence	Gold Standard	Best Guess
5	Roger **Clemens** to the Rangers .	NP	$(S[dcl]\backslash NP)/PP$
6	How many **combinations** are left?	N	$(N/S[inv])\backslash S[wq]$
7	Swan Song for **Hampton** ?	NP	$S[dcl]\backslash NP$
8	Is it rookie **Mercer** ?	NP	$S[dcl]\backslash S[q]$
9	Why has [CA] become **obsessed** with [NP]?	$(S[adj]\backslash NP)/PP$	$(S[dcl]\backslash S[wq])/PP$
10	**Winning** at all costs ?	$S[ng]\backslash NP$	$S[wq]/PP$

The most common set of errors can be characterised by the OOL target word being interpreted as the head of the sentence. The surrounding constituents are correct, but they are resolved as the arguments of the OOL category, the result being the sentence type motivated by the end punctuation.

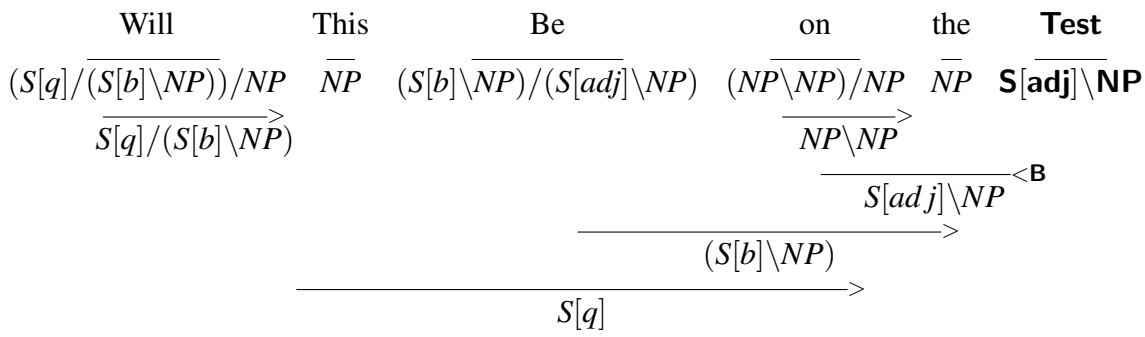
9.4.2.3 Gap realised as head, combined with misparses of common words

	Sentence	Gold Standard	Best Guess
11	Is there a **counterattack** ?	N	$S[dcl]\backslash S[q]$
12	Where were the **Newsweek** editors ?	NP	$(S[dcl]\backslash S[wq])/N$
13	Where are the **SAT** scores?	NP	$(S[dcl]\backslash S[dcl])/NP$

A third set of errors arises when the OOL word is resolved as the head of the sentences, but the other constituents are misattributed. This genuinely constitutes a separate type of error, as misinterpreting the OOL word as the head is a weakness of the learner, while the constituent errors are the fault of the baseline lexicon. In each of these three sentences, the determiner is given then category NP. It enables the learner to interpret the neighbouring constituents as full S types which the OOL word takes as arguments, rather than the correct nominal categories that are dependents of the determiner.

9.4.2.4 Remaining Errors

	Sentence	Gold Standard	Best Guess
14	Where else can [N] **entertain** clients ?	$(S[b]\backslash NP)/NP$	$(S[dcl]\backslash S[dcl])/NP$
15	**Send** in the clowns ?	$(S[imp]/(PP/NP))/NP$	NP
16	Will This Be on the **Test** ?	N	$S[adj]\backslash NP$
17	**Top** each side with chopped greens .	$(S[imp]/NP)/PP$	$S[adj]/S[adj]$
18	**Winning** national championships?	$(S[ng]\backslash NP)/NP$	$NP[nb]/N$



Of the 18 deemed to be erroneous, four are the fault of the CCGbank corpus, and 14 of the remaining 16 are found in sentences ending with a question mark. This indicates that question sentences are a major contributor to error in the learning process. Of 1080 newly parsed sentences, 73 contain question marks, or nearly 7%. By contrast, the evaluation set of §00 contains only 1% (18/1918) questions. This disparity indicates that our chosen training set is not representative of our test set, and by filtering out question sentences, we could eliminate a source of error and bring our corpora into line at the same time.

This error analysis shows us that a majority of errors arise from the variety and ambiguity of sentence types in Gigaword. If we had some dependable way to determine which strings of the corpus were $S[dcl]$, then these errors would be fewer. However, knowing the sentence type is made difficult by not knowing all the words of the sentence, so a post-processing step may be preferable, including only those lexical items that arise from sentences of whose result types we can be reasonably confident.

9.4.3 Responding to Lexical Errors

This lexical error analysis has shown that the two major contributors to error are new POS entries, and categories arising from question-sentences. We retrain the parser on the CI output, retroactively removing either the POS entries from the learned lexicon, or the question sentences from the training data. Removing the 73 question sentences from the CI training data results in only 86 new lexical entries, as opposed to 121, but of these, only four are erroneous, listed in table 9.9. This raises the lexical accuracy score from 85% to 95.3%.

Of the 59 entries lost, 24 are accounted for by POS entries, and the remaining 35 are lexical, consisting of 16 erroneous entries and 19 good ones. Removing the questions does away with all the bad POS entries. This is because new POS entries are

	W	C	Error Type
1	*Brunswick	$(N/N)/(N/N)$	Gold standard
2	*entries	$NP\backslash NP$	Gold standard
3	*Portland	$(N/N)/(N/N)$	Gold Standard
4	*Top	$S[adj]/S[adj]$	Lexical ambiguity

Table 9.9: Erroneous categories remaining after -Q question filter is applied.

created when new $w|c$ pairs are introduced, but aren't seen a sufficient number of times to meet the word-frequency threshold. They have to be truly rare for this to happen, and therefore more likely to be erroneous.

Filtering out the question sentences had the effect of eliminating all 24 of the new POS entries. The increase in F-score is accounted for by the increase in lexical category accuracy.

9.5 The Parser

In this section we examine the derivations in §00 yielded by the parser, before and after retraining on CI-learned sentences. First we present the precision and recall figures over dependencies, as compared to the gold standard parses. The analysis covers the characteristic errors and traces their origins to either the learned Gigaword sentences or the original training sentences. Parse results are the preferred method of evaluation in many standard lexicon-learning tasks, but here we treat them as a rigorous, but secondary, step in a deeper evaluation scheme.

9.5.1 Parser Results

Following on from the fine-grained analysis of the lexical results, we examine the parser's performance on category-match accuracy, defined as assigning a word its correct category as provided by the gold-standard corpus, without regard to the other words or structure in the sentence. These are shown for both test sections §00 and §23 in Tables 9.10 and 9.11 respectively.

Five lexical settings are compared in tables 9.10 and 9.11: the first row is the original StatCCG lexicon, trained on §00-21. It provides a very high baseline over both test sections. For comparison, we show the parser's performance with the POS-backoff disabled in the second row. This yields a higher precision figure, but much

	All Words			Target Set		
	P	R	F	P	R	F
BL	91.75	98.49	95.00	92.50	1	96.10
BL -BO	92.59	18.92	31.42	-	-	-
CI	91.60	98.19	94.78	91.42	1	95.52
CI -POS	91.61	98.19	94.79	91.48	1	95.55
CI -Q	91.75	98.49	95.00	92.40	1	96.05

Table 9.10: Category match performance over §00 for the baseline parser (BL) and the same with its POS-backoff mechanism disabled (BL-BO), compared to CI either with POS entries prohibited (CI-POS) or question sentences removed from training (CI-Q).

	All Words			Target Set		
	P	R	F	P	R	F
BL	91.81	97.58	94.61	90.52	1	95.02
BL -BO	93.08	24.01	38.17	-	-	-
CI	91.80	97.47	94.55	89.00	1	94.18
CI -POS	91.81	97.47	94.55	89.00	1	94.18
CI -Q	91.81	97.58	94.61	90.52	1	95.02

Table 9.11: Category match performance over §23 for the baseline parser (BL) and the same with its POS-backoff mechanism disabled (BL-BO), plain CI and the same either with POS entries prohibited (CI-POS) or question sentences removed from training (CI-Q).

	00			23		
	UP	UR	UF	UP	UR	UF
BL	89.79	89.11	89.45	89.87	88.18	89.02
CI -POS	89.65	88.74	89.19	89.86	88.06	88.95
CI -Q	89.77	89.12	89.44	89.85	88.16	89.00

Table 9.12: Unlabelled dependency recovery results for §00 and 23 after mining Gigaword for target words, as compared to the baseline before using Gigaword (BL).

	00			23		
	LP	LR	LF	LP	LR	LF
BL	82.71	82.09	82.40	83.17	81.60	82.38
CI -POS	82.53	81.69	82.11	83.18	81.50	82.33
CI -Q	82.69	82.09	82.39	83.15	81.59	82.36

Table 9.13: Labelled dependency results for §00 and 23 after mining Gigaword for target words, as compared to the baseline before using Gigaword (BL).

lower recall over all words, since the parser no longer has any mechanism for dealing with OOL words. Likewise, it scores 0 over our target set of OOL items, since it can attempt none of them.

The third row shows the results for the CI-retrained parser, which includes all 121 new entries discussed in the previous section. CI causes a slight drop in category match accuracy across the table, though the effect is more pronounced for the Target Set. Removing the 24 POS entries from the lexicon (CI -POS) has a small positive effect, while removing the questions (CI -Q) yields complete restoration of the F-score to baseline levels in all cases but one: §00's target set F-score still incurs a net loss of 0.05.

Even though category match figures are more informative from a lexical perspective, we also report standard head-dependency results for comparison. Table 9.12 shows the parsing performance of the retrained lexicon on §00, which it was tuned for, and §23, which it was not tuned for[2]. The former should show what impact the mining experiment has had on the lexicon as a whole. The latter should help us to

[2]A dependency analysis over the Target Set as in 9.10 and 9.11 may show more improvement, but the problem of teasing out just those dependencies that are relevant to those tokens is complex, though has potential for future development

		00			23		
		UP	UR	UF	UP	UR	UF
Long	BL	79.45	80.37	79.91	80.07	80.90	80.48
	CI -POS	78.90	80.31	79.60	79.95	80.68	80.31
	CI -Q	79.29	80.51	79.90	80.04	80.93	80.48
Short	BL	90.76	89.91	90.33	90.78	88.83	89.79
	CI -POS	90.66	89.51	90.08	90.79	88.72	89.74
	CI -Q	90.75	89.90	90.32	90.77	88.82	89.78

Table 9.14: Unlabelled long-range and short-range dependency results.

		00			23		
		LP	LR	LF	LP	LR	LF
Long	BL	66.97	67.75	67.36	68.77	69.56	69.16
	CI -POS	66.53	67.69	67.10	68.75	69.46	69.10
	CI -Q	66.84	67.90	67.37	68.76	69.61	69.18
Short	BL	84.18	83.40	83.79	84.51	82.69	83.59
	CI -POS	84.04	82.98	83.51	84.52	82.59	83.54
	CI -Q	84.17	83.39	83.78	84.50	82.67	83.57

Table 9.15: Labelled long-range and short-range dependency results.

discern the level of overgeneralisation incurred by mining for specific words. §23 is closed, and we do not know ahead of time how many of the OOL words in the Missing List it contains.

Because CI skips most sentences, there is a danger that the training data do not accurately reflect the test set, and that adding new short sentences with OOL words may improve short-range dependencies at the expense of long-range ones. There is a possibility that flooding the training data with short sentences will overfit, and as a result it will degrade performance on long sentences, since longer sentences tend to contain more long-range dependencies. However, Tables 9.14 and 9.15 show that long-range and short-range dependencies are equally affected by the addition of new training sentences, and learning from only short sentences does not damage the results.

As expected, the size of the lexicon has increased, while the parsing performance has not changed significantly. An analysis of the final lexicon has shown that 85% of the added lexical entries are plausible. This is an acceptable figure because the

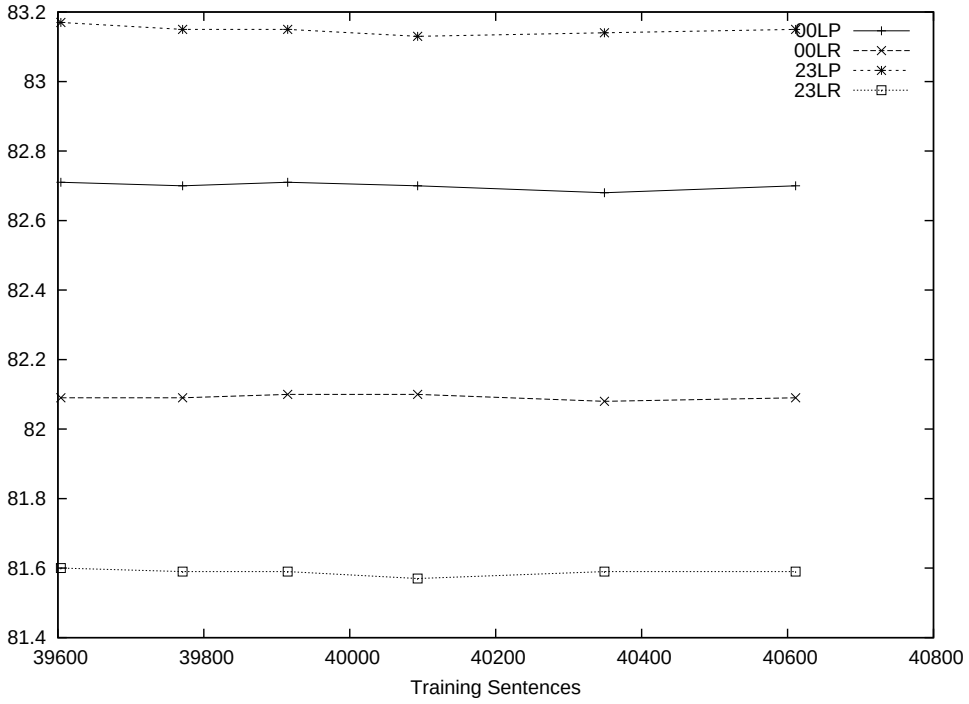


Figure 9.1: Comparison of labelled precision and recall for CI over dependencies in §00 and §23, as training sentences are added.

best-performing formulation of the StatCCG parser contains many erroneous lexical entries, but still yields state-of-the-art results. We rely on the robustness of the parser to smooth out small errors in the lexicon.

Figure 9.1 shows how precision and recall over dependencies are affected by the addition of the new training sentences. Each point in the series represents a training traversal of a whole NYT section. Performance stays relatively stable over the course of the learning process, for both §00 and 23.

Figures 9.2 and 9.3 compare long-range to non-long-range dependency performance on §00. Both values are quite stable as new sentences are added. This assuages the concern that imposing a length filter on sentences would bias the parser toward short-range dependencies to the detriment of long-range ones.

Figures 9.4 and 9.5 make the same comparison for §23.

9.5.2 Parser Analysis

In this section we examine the differences in parses between the baseline and CI-Q for §00 of CCGbank. Of the 1913 sentences of §00 only 102 sentences differ. Of these, only 20 have discrepancies in category assignments; the remaining 82 have only

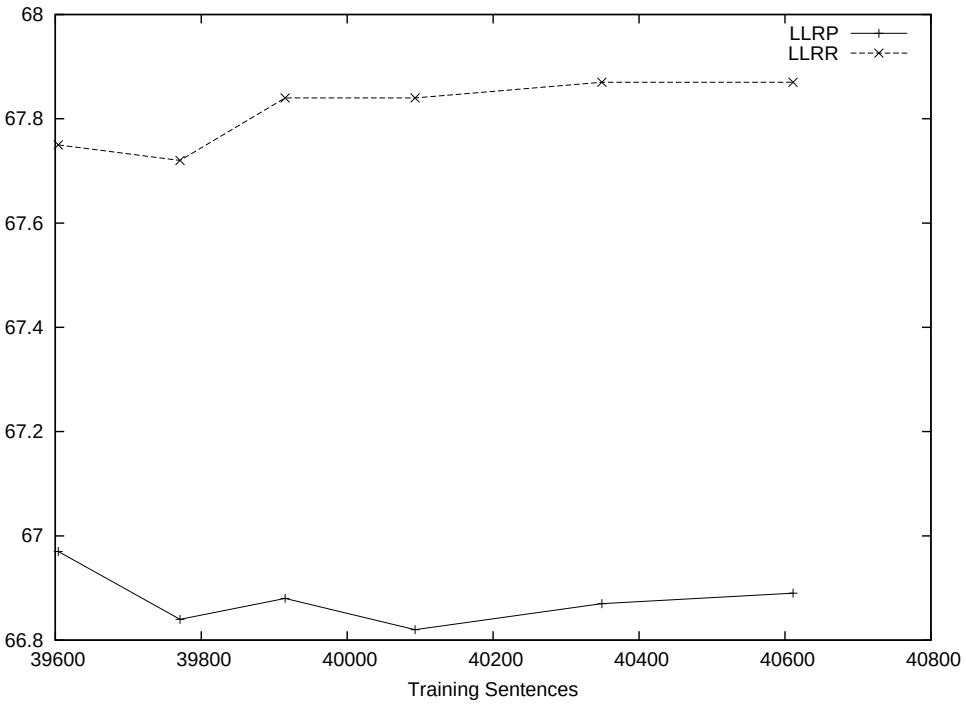


Figure 9.2: Labelled precision and recall for CI over long-range dependencies in §00 as training sentences are added.

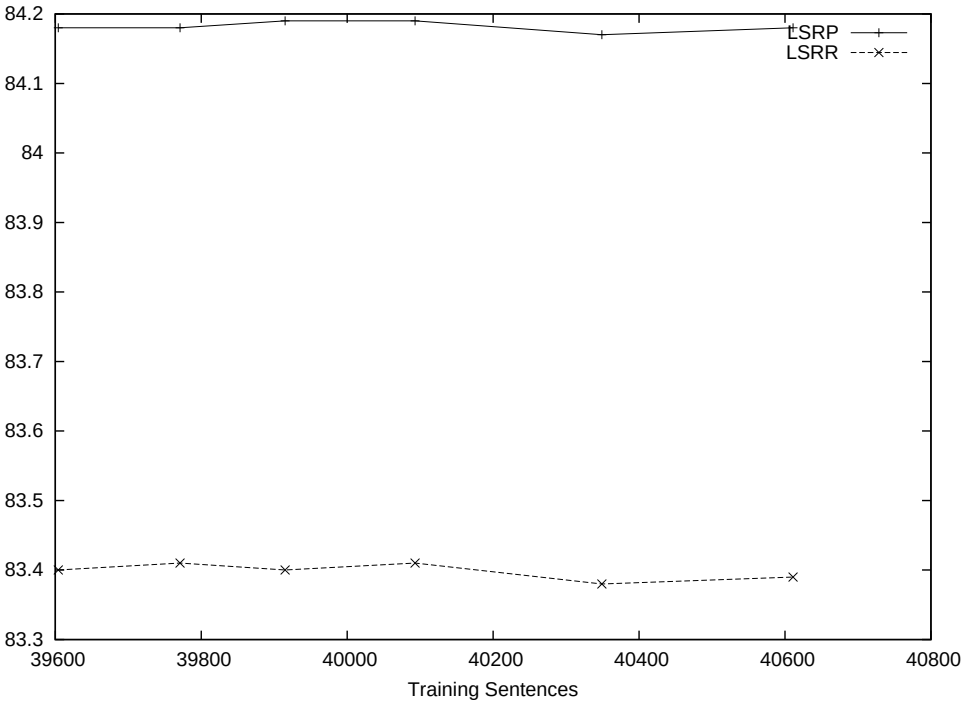


Figure 9.3: Labelled precision and recall for CI over non-long-range dependencies in §00 as training sentences are added.

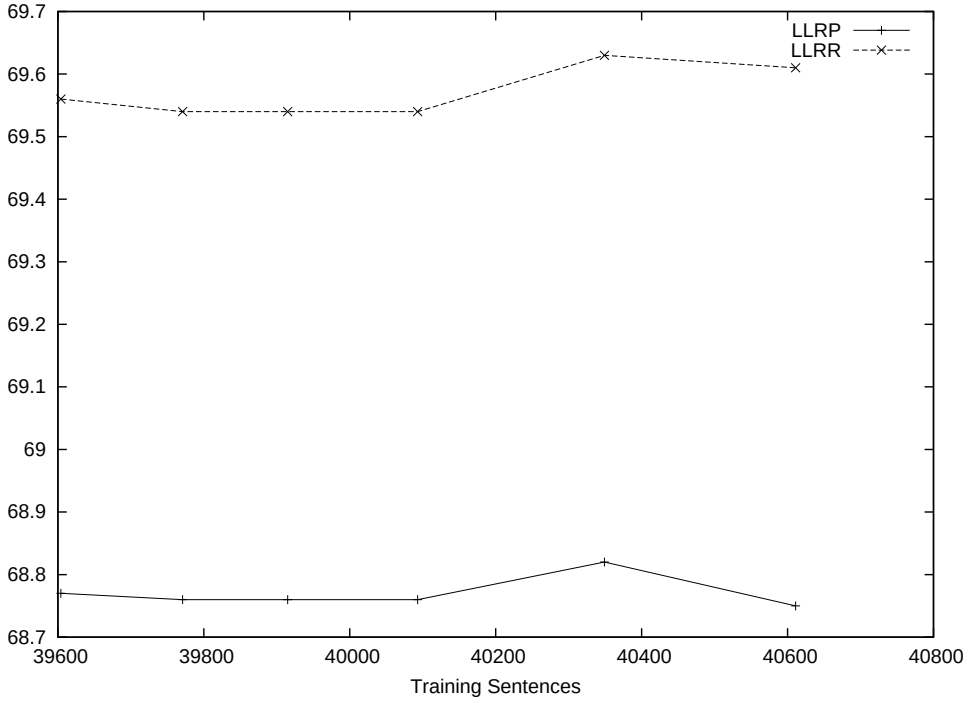


Figure 9.4: Labelled precision and recall for CI over long-range dependencies in §23 as training sentences are added.

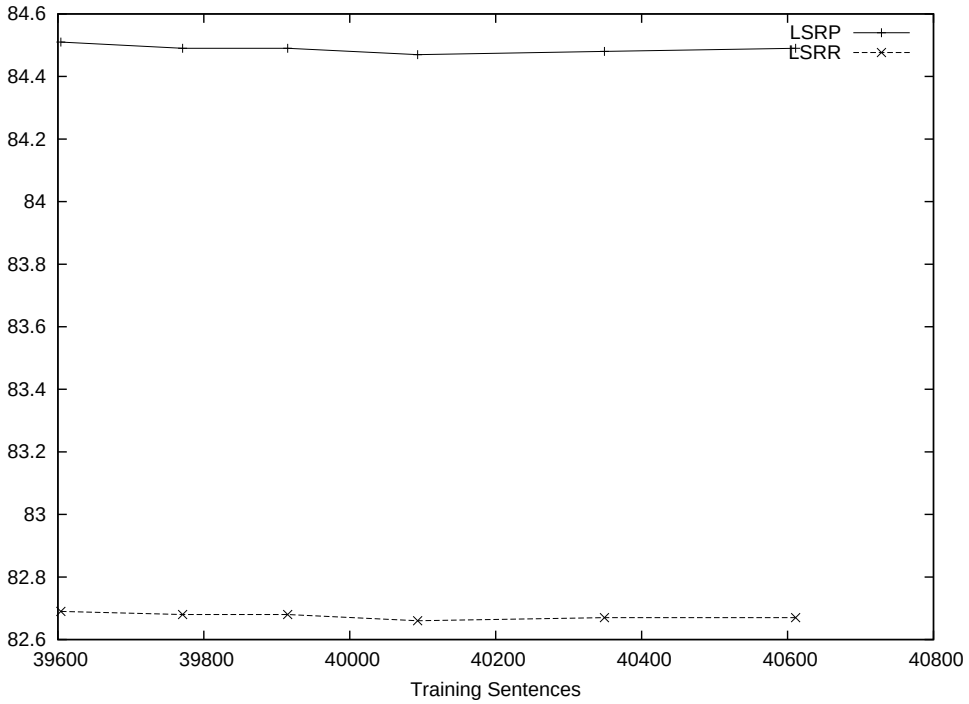


Figure 9.5: Labelled precision and recall for CI over non-long-range dependencies in §23 as training sentences are added.

structural differences.

Analysis of these 20 sentences yields 28 unrelated "incidents": interrelated structures where an independence assumption cannot be made. They involve 49 word tokens. These discrepancies are listed in table 9.16 and continued in 9.17. A manual review of the word/category pairs reveals that the Baseline sentences contain 19 starred items, and the CI-Q sentences 25, for a net loss of 6. This helps to explain the 0.05 difference in F-score on category matches above. In some cases the baseline or CI-Q categories were deemed equivalent to the gold standard, and in one case the gold standard itself was deemed to be in error.

9.6 The Learner

In this section we examine the new Gigaword parses produced by the learner. By tracing the CI-motivated errors that led to the parse errors in the previous section, we can typify the learning strategies that result in the best possible learning.

Of the ten million sentences in our subset of Gigaword, the learner only produces parses for 1107. Most sentences are timed out. We estimate that any sentence complex enough to time out will be likely to yield a worse result than less complex sentences, so we are happy to let them go for precision reasons, even though they take up much of the run time. In addition, not every category set yields a parsed sentence. Since we are using the hypothesis parser to produce a full derivation for every attempted sentence, sometimes this fails and the time used to produce the solution can be considered wasted.

9.6.1 Learner Results

1107 new parsed sentences were produced, after filtering for questions; we take a 10% sample and review them manually. This sample of 100 sentences includes only 82 unique parses. Table 9.18 shows the whole-parse precision for the 12 different sentence types produced by the learner. It shows that the sentence type we think of as most common ($S[dcl]$) is underrepresented by the learner. This is motivated by the end punctuation model, which allows a variety of sentence results, weighted by their representation in the training corpus. As we see here, $S[dcl]$ should be given more probability mass at the expense of the others. This also suggests that a post-facto filter that only allows $S[dcl]$ parses into the retraining corpus may be productive.

	W	Gold	Baseline	NoQ
1	dust	N	N	N/N
	that	$(NP\backslash NP)/(S[dcl]\backslash NP)$	$(NP\backslash NP)/(S[dcl]\backslash NP)$	$*N/N$
	hung	$S[dcl]\backslash NP$	$(S[dcl]\backslash NP)/PP$	$*N/N$
	over	$((S\backslash NP)\backslash(S\backslash NP))/NP$	PP/NP	$*N/N$
2	viewpoint	N	$*N/N$	N
3	a	$NP[nb]/N$	$*N$	$*NP$
4	October	N/N	$*N$	N/N
	survey	N	$*(S[dcl]\backslash NP)/NP$	N
5	on	PP/NP	$(NP\backslash NP)/NP$	$*(S\backslash NP)/NP$
6	been	$(S[pt]\backslash NP)/(S[adj]\backslash NP)$	$(S[pt]\backslash NP)/(S[ng]\backslash NP)$	$*((S[pt]\backslash NP)/(S[to]\backslash NP))/(S[adj]\backslash NP)$
	enormously	$(S[adj]\backslash NP)/(S[adj]\backslash NP)$	$(S\backslash NP)/(S\backslash NP)$	$(S[adj]\backslash NP)/(S[adj]\backslash NP)$
	frustrating	$(S[adj]\backslash NP)/PP$	$(S[ng]\backslash NP)/PP$	$S[adj]\backslash NP$
	to	PP/NP	PP/NP	$*((S[adj]\backslash NP)\backslash(S[adj]\backslash NP))/NP$
7	raring	$(S[adj]\backslash NP)/(S[to]\backslash NP)$	$(S[adj]\backslash NP)/(S[to]\backslash NP)$	$*S[adj]\backslash NP$
8	shrinks	$S[dcl]\backslash NP$	$*(S[dcl]\backslash NP)/S[dcl]$	$S[dcl]\backslash NP$
	while	$((S\backslash NP)\backslash(S\backslash NP))/S[dcl]$	$*((S\backslash NP)\backslash(S\backslash NP))/NP$	$((S\backslash NP)\backslash(S\backslash NP))/S[dcl]$
9	's	$*N$	$*N$	$NP\backslash NP$
10	one	N	N	$*(NP\backslash NP)/(NP\backslash NP)$
11	in	$((S\backslash NP)\backslash(S\backslash NP))/NP$	$*(S\backslash NP)/NP$	$((S\backslash NP)\backslash(S\backslash NP))/NP$
12	close	N/N	N/N	$*NP[nb]/N$
13	laughing	N/N	N/N	$S[ng]\backslash NP$
14	laughing	$S[ng]\backslash NP$	$*(S[ng]\backslash NP)/(S[adj]\backslash NP)$	$S[ng]\backslash NP$
	much	$(S\backslash NP)\backslash(S\backslash NP)$	$*S[adj]\backslash NP$	$(S\backslash NP)\backslash(S\backslash NP)$
15	after	$((S\backslash NP)\backslash(S\backslash NP))/S[dcl]$	$((S\backslash NP)\backslash(S\backslash NP))/S[dcl]$	$*((S\backslash NP)\backslash(S\backslash NP))/NP$
16	on	$(NP\backslash NP)/NP$	$*((S\backslash NP)\backslash(S\backslash NP))/NP$	$(NP\backslash NP)/NP$

Table 9.16: Parse discrepancies over §00 (continued on next page).

	W	Gold	Baseline	NoQ
17	is	$(S[dcl]\backslash NP)/(S[adj]\backslash NP)$	$(S[dcl]\backslash NP)/(S[adj]\backslash NP)$	$*(S[dcl]\backslash NP)/S[dcl]$
	furious	$(S[adj]\backslash NP)/S[dcl]$	$(S[adj]\backslash NP)/S[dcl]$	N/N
	its	$NP[nb]/N$	$NP[nb]/N$	$*N/N$
18	plus	$(NP\backslash NP)/(NP\backslash NP)$	$(NP\backslash NP)/NP$	$conj$
19	Like	PP/NP	PP/NP	$(S/S)/NP$
20	Think	$(S[b]\backslash NP)/PP$	$(S[b]\backslash NP)/PP$	$*(S[dcl]\backslash NP)/S[dcl]$
	about	PP/NP	PP/NP	$*((S\backslash NP)\backslash(S\backslash NP))/NP$
21	do	$((S[b]\backslash NP)/PP)/NP$	$*(S[b]\backslash NP)/NP$	$*S[b]\backslash NP$
	for	PP/NP	$(NP\backslash NP)/NP$	$((S\backslash NP)\backslash(S\backslash NP))/NP$
22	about	NP/NP	$*(NP\backslash NP)/NP$	$*((S\backslash NP)\backslash(S\backslash NP))/NP$
23	sounded	$S[dcl]\backslash NP$	$S[dcl]\backslash NP$	$*(S[dcl]\backslash NP)/(S[adj]\backslash NP)$
	here	$(S\backslash NP)\backslash(S\backslash NP)$	$(S\backslash NP)\backslash(S\backslash NP)$	$*S[adj]\backslash NP$
24	've	$(S[dcl]\backslash NP)/(S[pt]\backslash NP)$	$(S[dcl]\backslash NP)/(S[pt]\backslash NP)$	$*(S[b]\backslash NP)/(S[pt]\backslash NP)$
25	just	$(S\backslash NP)/(S\backslash NP)$	$*((S\backslash NP)\backslash(S\backslash NP))/((S\backslash NP)\backslash(S\backslash NP))$	$(S\backslash NP)/(S\backslash NP)$
	drift	$S[dcl]\backslash NP$	$*(S\backslash NP)\backslash(S\backslash NP)$	$(S[b]\backslash NP)/(S[adj]\backslash NP)$
	away	$(S\backslash NP)\backslash(S\backslash NP)$	$(S\backslash NP)\backslash(S\backslash NP)$	$S[adj]\backslash NP$
26	When	$(S/S)/S[dcl]$	$*(S/S)/S[pss]$	$(S/S)/S[dcl]$
	established	$(S[dcl]\backslash NP)/S[em]$	$*S[pss]\backslash NP$	$*(S[dcl]\backslash NP)/NP$
	that	$S[em]/S[dcl]$	$((S\backslash NP)\backslash(S\backslash NP))/S[dcl]$	$*NP$
27	caveat	N	N	$*N/N$
	was	$(S[dcl]\backslash NP)/(S[pss]\backslash NP)$	$(S[dcl]\backslash NP)/(S[pss]\backslash NP)$	$*S[dcl]\backslash NP$
	registered	$S[pss]\backslash NP$	$S[pss]\backslash NP$	N/N
28	Sure	$(S/S)/(S/S)$	$*S[adj]\backslash NP$	$*S/S$
	enough	S/S	$*(S[adj]\backslash NP)\backslash(S[adj]\backslash NP)$	S/S

Table 9.17: Parse discrepancies over §00, cont.

R	Count	Correct
$S[dcl]$	39	10
$S[b]$	15	0
S	7	0
$S[adj]$	5	0
$(S/S)/(S/S)$	4	1
$NP[nb]$	3	0
$S[ng]$	3	0
$S[pt]$	2	0
N	1	0
N/N	1	0
PP	1	0
$(S[b]\backslash NP)\backslash(S[b]\backslash NP)$	1	0
Total	82	11(13.4%)

Table 9.18: Whole-parse precision of 10% sample of CI parses, by sentence result type (R).

Of the 361 individual word tokens in these 82 sentences, 113 were manually determined to be errors, resulting in 69% category accuracy over the Gigaword sample. This is lower than the category match results on CCGbank, which may be due to the different domain of the sentences. On average, there were 1.38 errors per sentence over all sentences, and 1.59 errors per error-containing sentence, which shows that there is a tendency for one bad category in a derivation to lead to another. The most common errors included determiners and prepositions. Eleven determiners, all with POS-tag DT, were given an incorrect category, as were six prepositions, all with POS-tag IN. Prepositions were loosely graded; any prepositional category that afforded the correct attachment was deemed correct[3].

9.6.2 Learner Analysis

Below we examine some of the most characteristic errors in detail. The sentences 'Two color slides .' and 'Three color slides .' occur a total of 16 times in the sample, due to their high frequency as a journalistic device in Gigaword[4]. However, they are frag-

[3]Prepositional categories are defined as PP, $(S\backslash NP)\backslash(S\backslash NP)$, $NP\backslash NP$, $S\backslash S$ and S/S.

[4]Four more sentences in the Gigaword sample occur twice.

Corpus	Sentences Added	$\|lex\|$	Items Added
baseline	-	38054	-
nyt10	167	38064	10
nyt11	144	38087	23
nyt12	178	38106	19
nyt13	256	38125	19
nyt14	262	38138	13
All	1007	38138	84

Table 9.19: Number of labelled sentences and lexical items added as more sections of Gigaword are added to the training set.

ments, and therefore easy traps for CI to get itself into. In this case, *slides* is the OOL word, and CI has determined that it is an intransitive verb. This is not a catastrophic error, because *slides* occurs as an intransitive in other contexts. Unfortunately, it has also decided that it is a bare verb ([b]), rather than a declarative ([dcl]). The adjectival form of *color* is present in the baseline lexicon, so it is available to the learner, but unused.

(15) * Two|Three color **slides**

$\overline{N/N}$ $\overline{N}$ $\mathbf{S}\overline{[\mathbf{b}]\backslash\mathbf{NP}}$

$\dfrac{}{N}>$

$\dfrac{}{NP}^{M}$

$\dfrac{}{S[b]}<$

(16) Two|Three color **slides**

$\overline{N[num]}$ $\overline{N}$ $\overline{\mathbf{N}}$

$\dfrac{}{N/N}^{M}$ $\dfrac{}{N/N}^{M}$

$\dfrac{}{N}>$

$\dfrac{}{N}>$

The second type of common error is the sentence-modifier error. In the sentence below, *They shoot* has been realised as a sentence, using the valid but inappropriate intransitive sense of *shoot*. The OOL word *arrows* is at the edge of the sentence, and CI exploits this position to assign it the category that turns its sentential neighbour into a declarative sentence.

(17) * They shoot **arrows**

$\overline{NP}$ $S[\overline{b}]\backslash NP$ $\mathbf{S}[\overline{\mathbf{dcl}}]\backslash\mathbf{S}[\mathbf{b}]$

$\overline{S[b]}\,<$

$\overline{S[dcl]}\,<$

(18) They shoot **arrows**

$\overline{NP}$ $(S[\overline{dcl}]\backslash NP)/NP$ $\overline{\mathbf{NP}}$

$\overline{S[dcl]\backslash NP}\,>$

$\overline{S[dcl]}\,<$

Attachment errors are also rife: here, *Tuscany* is the OOL word, and, while it gets the nominal role correct, it erroneously includes the adjunct *PP* as an argument.

(19) * Take **Tuscany** as an example

$(S[b]\backslash\overline{NP})/NP$ $\overline{\mathbf{NP}}/\overline{\mathbf{PP}}$ $\overline{PP}/\overline{NP}$ $\overline{NP[nb]}/\overline{N}$ $\overline{N}$

$\overline{NP[nb]}\,>$

$\overline{PP}\,>$

$\overline{NP}\,>$

$\overline{S[b]\backslash NP}\,>$

(20) Take **Tuscany** as an example

$S[\overline{imp}]/NP$ $\overline{\mathbf{NP}}$ $\overline{PP}/\overline{NP}$ $\overline{NP[nb]}/\overline{N}$ $\overline{N}$

$\overline{S[imp]}\,>$

$\overline{NP[nb]}\,>$

$\overline{PP}\,>$

$\overline{\qquad}\,M$

$S\backslash S$

$\overline{S[imp]}\,<$

The most linguistically interesting, and typical, sentence is 'Varying cover charges.' Here, each word is a homonym, having a number of different entries in the lexicon: *cover* has four and *charges* ten, listed in figure 9.20. Recovery of the OOL word *Varying* is therefore fraught with ambiguity, and the learner relies on the probabilities to choose among all the options. In this case, *NP* wins, making a neat, but incorrect, transitive sentence.

(21) * **Varying** cover charges

$\overline{\mathbf{NP}}$ $(S[\overline{b}]\backslash NP)/NP$ $\overline{N}$

$\overline{S[b]\backslash NP}\,>$

$\overline{S[b]}\,<$

(22) * **Varying** cover charges

$$\dfrac{\dfrac{\dfrac{\overline{\mathbf{N}/\mathbf{N}}}{N}>}{NP}M \qquad \overline{N} \qquad (S[\overline{dcl}]\backslash NP)/NP}{S[dcl]\backslash NP}<$$

(23) * **Varying** cover charges

$$\dfrac{\dfrac{\overline{\mathbf{N}/\mathbf{N}} \qquad \overline{N/N}}{N/N}>\mathbf{B} \qquad N/\overline{S[dcl]}}{N/S[dcl]}>\mathbf{B}$$

(24) **Varying** cover charges

$$\dfrac{\overline{\mathbf{N}/\mathbf{N}} \qquad \dfrac{\overline{N/N} \qquad \overline{N}}{N/N}>}{N}< \Big)M$$
$$NP$$

(22) and (23) would yield the correct category for the OOL word, even though the parse is incorrect. The intransitive sense for *charges* is missing from baseline lexicon, but the intransitive sense yields a non-sentential result of $S[dcl]\backslash NP$, which is a lower probability. The two highest-probability categories for *charges*, $N/S[dcl]$[5] and $N/[em]$[6] yield low-probability sentential results, and so are dispreferred. (24) is the correct parse, to which the learner has access, but skips over in favour of the higher marginal likelihood of (21).

Analysis shows that $S[dcl]$ types are most reliable, and other result types are over-represented. Determiners and prepositions are misparsed more often than anticipated. This suggests that a more curated lexicon would lead to better CI parses, since it would not allow these kinds of errors to occur. The same old generative bias is at work. Certain sentences occur identically or near-identically in the corpus, and are therefore overrepresented. This is also taking up time and resources from more interesting/informative sentences.

[5] 1 instance of w/c pair *charges*:=$N/S[dcl]$ occurring in CCGbank " ...plead guilty ... to *charges* it illegally steered company money to politicians ..."; 31 other lexical entries share this category.

[6] 13 instances of w/c pair *charges*:=$N/S[em]$ in CCGbank, e.g. " ... fighting *charges* that it bribed Jamaican officials ..."

W	C	$P(W\|C)$
cover	$(S[b]\backslash NP)/NP$	0.004296785084017494
cover	$(S[dcl]\backslash NP)/NP$	1.8685767673621924E-4
cover	N	1.93881112102059E-5
cover	N/N	6.557033073674824E-6
charges	$N/S[dcl]$	0.018518518518518517
charges	$N/S[em]$	0.017793594306049824
charges	$((S[dcl]\backslash NP)/NP)/NP$	0.005509641873278237
charges	$(S[dcl]\backslash NP)/S[em]$	0.002190100744634253
charges	$((S[dcl]\backslash NP)/PP)/NP$	0.001854140914709518
charges	$(S[dcl]\backslash S[dcl])\backslash NP$	0.0011013215859030838
charges	N	6.83430920159758E-4
charges	$(S[dcl]\backslash S[dcl])/NP$	6.46830530401035E-4
charges	$(S[dcl]\backslash NP)/S[dcl]$	3.5733428622476324E-4
charges	$(S[dcl]\backslash NP)/NP$	1.2457178449081283E-4

Table 9.20: Entries for *cover* and *charges* in the baseline lexicon.

9.7 Discussion

In this experiment, we have tested the performance of CI in recovering the 2543 word-category pairs in §00 of CCGbank that are not covered by the baseline lexicon. The result is that 86, or 3.4% of the target set, are recovered. An in-depth analysis of all the stages of lexical discovery shows that the 86 learned lexical items represent an improvement over the seed lexicon. However, we find that this small lexical improvements are not reflected in the parser. Zhang et al. [2005; 2007] noted this as well, finding that lexical evaluation does not strongly correlate with parsing performance. He suggests weighting lexical F-scores more in favour of recall, as it is a better indicator of parsing ability than is precision.

Removing questions from the training corpus is an effective strategy for increasing the precision of the learner. This is due to the baseline CCGbank corpus having inadequate support for questions, as discussed in chapter 8, which is driven by the low frequency of questions in the NYT sections of the Penn Treebank. For different combinations of corpora, there might be a different set of sentence types that are underserved, so future users of CI should take care to evaluate any domain issues when setting their parameters.

The final step is to estimate how many Gigaword sentences would need to be put to the learner in order for all 2543 lexical items to be learned. This experiment shows that only 1107 sentences out of the available 10 million were capable of contributing to word learning, or about 200 sentences for each section of the NYT. We project that running CI over all 23 available sections of the NYT would yield just over 4600 learnable sentences. Since the NYT section, representing 48,000,000 sentences, is only half of the Gigaword corpus, we estimate that the whole set would make available only around 9000 useful sentences.

At a rate of 86 new words for every 1007 sentences, given the current speed of the implementation and its timeout settings, we project that these 9000 sentences would result in about 769 new lexical entries, which comes nowhere near to the 2543 needed to handle §00. This estimate does not take into account the acceleration of learning that occurs once the word frequency threshold is passed, or the deceleration that results as we go further into the long tail, but we expect these two elements to balance each other out in the long run. With improved implementational efficiency and more, possibly parallel, computing power, this task could be achieved over time, but should not necessarily be attempted. While we have made the recovery of a set of known word-category pairs the goal of this experiment, it does not directly translate to the real-world task of lexical acquisition for the improvement of parser coverage.

The generative model seeks to maximise the probability over the derivation. Using the $P(word|category)$ means that, all other things being equal, the maximised choice is the rarer entry. If the choice is between a common category (whose probability mass is spread among a large number of word types), and a rare category (where this word has a large proportion of the probability mass), the latter will be preferred. This is simply a property of generative parsing, and the framework under which normal parsing takes place is suitably robust to this. However, when using these lexical rules to identify OOL words, the system is vulnerable to this false economy. Lexical smoothing in the vein of [Deoskar 2009] is a potential avenue for future work.

The lexical approach to this problem is to use a stricter method of adding items to the lexicon. StatCCG's method is to raise the word frequency threshold, but that still allows singleton pairs to enter the lexicon. Another option would be to keep track of the counts as well as the probabilities of each lexical entry, then use these numbers as an additional metric, or element in the maximisation equation.

Inasmuch as lexical bias is a weakness of generative parsing, it is also a weakness of the CI algorithm. The same weakness plagues POS backoff, BF, and any method of

word learning that takes advantage of generative lexical probabilities.

Given more time, the experiment would benefit from further filtering the learner to ignore questions, weighting the bias to S[dcl] more heavily (which must be done artificially), and pruning the lexicon to only those w/c pairs that are stable and useful. This brings up the same paradox has been discussed previously: rare categories will be found in rare contexts, and one rare category leads to another. POS information is required by the parser, so keeping that information away from the learner makes the learning problem simpler to implement, but unnecessarily data-poor. A hybrid model that takes both POS and structure into account, in the vein of Auli and Lopez [2011]; Koo et al. [2010], would be ideal. Ideally, a more efficient implementation of the parser will be made available for future use. We also expect that the new version of StatOpenCCG will have solved the timeout issue, which caused the parser to spend too long failing on certain sentences despite a timeout parameter.

In conclusion, Chart Inference succeeds in widening lexical coverage over the StatCCG seed lexicon when allowed to learn new lexical items from the Gigaword corpus. In addition to learning more valid entries than spurious ones, the parsing results as compared to the baseline are no worse, and the error analysis shows interesting linguistic phenomena that will help to influence future incarnations of CI. The application of filters and restrictive parameters are instrumental in weeding out sentences that are likely to be problematic. Similarly, it is vital that the learning corpus accurately reflect the test sentences, as is shown when questions are filtered out from the learning set.

Chapter 10

Conclusion and Future Work

10.1 Summary

This thesis has presented a semi-supervised word learning system that employs a chart to generate CCG category types for unknown words. The first two chapters are turned over to background and review of previous work. The remainder of the work is involved in the testing of CI in word-learning tasks of increasing complexity.

Chapter 3 describes two baseline systems, Brute Force (BF) and Rule Based (RB), and sets out the extensions to them that we have achieved in the implementation. Chapter 4 introduces Chart Inference (CI), as a faster and more accurate solution to the same word-learning problem. Chapter 5 shows that CI and RB converge in the same way as established by BF, at the same time indicting that the size of a seed is not as important as its quality, and that cosine similarity over a $P(w|c)$ lexical model is not sufficient for evaluating the quality of a lexicon.

Chapter 6 introduces the McGuffey corpus as a CCG resource for testing language learning systems. It shows that CI outperforms BF and RB on practical natural language tasks. Analysis also shows that CI, as an algorithm, is capable of covering all natural categorial types, and does not suffer from any systematic deficiencies, in contrast to the BR and RB baselines. The McGuffey corpus will be made available as a CCG resource for research into language acquisition, both human and artificial. Its compatibility with CCGbank and its increasing order of linguistic complexity make it a potentially valuable corpus for learning systems.

Chapter 7 tested CI in a larger-scale setting, in the task of recovering as many lexical items as possible from the CCGbank corpus, given various sizes of seed corpus. CI was shown to be superior to a self-training (ST) baseline that uses the parser's built-

in POS backoff mechanism to deal with unknown words, in that it produces fewer, higher-quality parses over the same data.

Chapter 8 is a domain-adaptation task, attempting to learn missing types for question-word categories, from which we draw the conclusion that CI succeeds in finding new category types, but that it is too slow and restrictive to make a large impact on the lexical model. Using CI in combination with ST provides the best results, an improvement of 50% in precision over the target set of SWHE questions.

Chapter 9 undertakes the most arduous learning task, and shows that CI can play a role in discovering OOL lexical items, but also that the initial conditions heavily influence the items learned, and that the model is subject to vagaries of the implementation. We report results of 121 new lexical items learned with 85% accuracy, rising to 95% accuracy for 86 new items when we remove question sentences from the training corpus. The extended lexicon does not afford an appreciable change in F-score over dependencies when used to parse the test set, highlighting the difficulty in evaluating lexicon quality at a parser level. We also execute a power analysis that suggests that a word learner would need orders of magnitude more unlabelled training data to cover all the missing words from §00.

10.2 Contributions

The central novel contributions of this thesis are threefold: the statistical extension to Yao's rule-based learner, the creation of the McGuffey corpus, and the development of Chart Inference (CI) as a solution to the word-learning challenge introduced by Brent. This thesis introduces a new method for discovering CCG category types for out-of-lexicon words. Chart Inference (CI) takes advantage of a pre-trained model and expands the lexicon using unlabelled text. It is capable of generating new category types, and is therefore more appropriate than backoff via Part-of-Speech for learning new lexical items.

CI is particularly useful for boosting a treebank with rare constructions, and we have shown that it does not produce the catastrophic results to which ST is prone. It is a powerful method for combining the efficient language model gained from labelled data with the variety and abundance of unlabelled data to broaden the coverage of an existing lexicon. In terms of real-world NLP tasks, CI has shown promise in the fields of lexicon acquisition starting with a small amount of linguistic knowledge (chapter 5), adapting existing wide-coverage lexicons to new syntactic domains (chapter 8),

and extending the coverage of treebank-trained parser with no additional labelled data (chapter 9.)

10.3 Future Work

This thesis represents the first efforts to implement CI and deploy it in natural language tasks, but in order to compete with the state-of-the-art, more work will be necessary.

Efficiency

The next logical step is to apply CI to a truly large-scale learning problem. There are problems of parsing efficiency and pruning to overcome in order to make this undertaking time-efficient, but this thesis shows that CI is capable of finding good-quality lexical items that are otherwise missing from a very good treebank. If the implementation were sped up, more data could be processed in less time. We have experienced problems with the StatOpenCCG timeout mechanism. An updated version of StatOpenCCG has been released since the completion of this thesis, and has the potential to introduce improvements across the learning pipeline.

Power

Our CI implementation does not cover the full generative power of CCG, because it only uses application and composition. However, the crossed composition and substitution combinators are much rarer, and most of their behaviour is encoded in the model rules, to which CI has full access. Therefore, our system has the best of both worlds, since it cannot over-apply these rare rules, but can still take advantage of the truly justified ones. Potential future projects would extend CI to cover the remaining combinators, as well as the multi-modal CCG extensions, and test whether they improve the algorithm.

Chart revision

We have not yet fully investigated the reinterpretation power of CI over the partial chart. Initial investigation showed no increase in precision when we allowed reanalysis of filled cells, but processing time rose considerably. In light of this, all experiments

were run with filled cells locked, meaning that any cell that was filled by the bottom-up partial parsing process cannot also accept new entries from the top-down CI process.

This decision was also motivated by the incommensurability of the bottom-up and top-down probability calculations. We expect that for languages with fewer available resources, and therefore smaller seed lexicons, it may be profitable to allow reanalysis of filled cells, so that CI might have a chance to recover from malformed constituents.

Known Words

This thesis addresses the OOL problem, but only in the case of completely unseen words. It can find both known and unknown categories for unknown words, but the problem of how to find additional categories for known words remains. This is a problem faced by all systems attempting to extend the coverage of a lexicon [Deoskar 2009], and is extremely difficult. It is related to the problem of when to stop learning, and we skirt the issue by compiling the learned sentences into the second-level training data. It is a problem for ST, too, and in many domains other than lexicon acquisition.

The ideal method for discovering gaps in an existing lexicon is elusive, and therefore a rich area for further work. The method used in chapter 5 was to reparse every sentence for which a parse was impossible, taking each word in turn as an unknown. This approach could be extended to sentences for which a parse is possible, and used to determine whether any of the learned sentences produce a higher-probability parse than the original. However, in its limited formulation it was found to be too time-consuming to be applied to a large-scale natural language task. We address this problem partially in the QA domain adaptation experiments in chapter 8, but the difference is that we know which words we have incomplete information for, and go out to mine the available data to boost our knowledge of them. In this sense, it is reasonable to use CI (or any number of semi-supervised methods) to learn more about known words, but the problem of determining for which words unseen categories exist, remains.

Languages

A pertinent avenue of future investigation is to implement Chart Inference for a language other than English. Some preliminary work has been done with Thai, and the integration was relatively simple. One only needs a parser that is capable of reporting partial parses, and a method for interpreting parser output and formatting data for input. A flexible lexical model is also necessary, but the learning could be done in a

single pass, as in chapters 6-9, or incrementally, as in chapter 5. The latter would be a better choice if starting with a very small lexicon.

Grammars

Although CI was developed for use with CCG, we anticipate that it could be adapted to work for various grammar formalisms. Since it employs two methods for generating cells in the chart (inverse combinators and model rules), it could be easily modified to use model rules only, and as such, use the rules of any model. The chart and the algorithm would be the same; only the representation would differ. However, this would be an incomplete method for learning a grammar from a very small seed, since most other grammar formalisms have distinct lexical and rule components. CCG's particular strength in this respect is that the universal combinators allow for constituent combinations that have not been attested in the training set. DLA for HPSG is more complex than our tasks, since the architecture of CCG conflates learning the rules of a language with learning the lexicon.

Morphology

Morphology is a rich and informative area of inquiry, and we foresee CI being integrated with a morphological model, most simply with the POS tags that must be attached to the learning corpus for StatCCG compatibility, or with a suffix analyser. Suffix information may not be available in all cases, but the input from a POS tagger would help to limit the set of categories CI would consider. At the same time, it would be a barrier to innovation, because the POS tagger would presumably be trained on the same corpus as the initial parser, which we know to be insufficient. A fruitful integration of morphology into the word learning mechanism would take this into account.

10.4 Applications

There are two areas of application in which CI has the potential to excel. The first is in semi-automatically discovering a grammar for a resource-poor language for which a small, precise seed lexicon is manually created: CI could serve as a tool for lexicographers creating new CCG corpora and parsers. Some work has been done in this direction, using CI to iteratively build a lexicon for Thai. CI could be a useful part of a lexicon acquisition effort that makes efficient use of a human speaker's time to create

a seed and check the output parses, in combination with computational power to find the generalisable patterns in the language. The challenges to this task include creating a high-quality seed, and ensuring that the atomic categories cover the needs of the language, as well as focusing the training sentences to build up the lexicon incrementally, in the style of the McGuffey corpus.

The second application is in extending lexicons, and thereby providing more labelled data, for state-of-the-art parsers, to increase their coverage from raw text when labelled resources are exhausted. The latter task garners more attention from the scientific community, in particular because it is so difficult: as we have seen in chapter 9, local improvements to the lexicon do not immediately lead to appreciable increases in F-score. However, the preliminary work of this thesis into the field of lexicon extension shows that Chart Inference has the potential to contribute to the solution.

Bibliography

Auli, M. and Lopez, A. (2011). A comparison of loopy belief propagation and dual decomposition for integrated CCG supertagging and parsing. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies*, pages 470–480, Portland, OR. ACL.

Baldridge, J. and Kruijff, G.-J. M. (2003). Multi-modal combinatory categorial grammar. In *Proceedings of the tenth conference on European chapter of the ACL - Volume 1*, EACL '03, pages 211–218, Stroudsburg, PA, USA. ACL.

Baldwin, T. (2005). Bootstrapping deep lexical resources: Resources for courses. In *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, pages 67–76, Ann Arbor, Michigan. ACL.

Baldwin, T., Korhonen, A., and Villavicencio, A., editors (2005). *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*. ACL, Ann Arbor, Michigan.

Bangalore, S. and Joshi, A. K. (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.

Barg, P. and Walther, M. (1998). Processing unknown words in HPSG. In *Proceedings of the 36th Conference of the ACL and the 17th International Conference on Computational Linguistics*, volume cs.CL/9809106, pages 91–95, Montreal, Canada.

Blunsom, P. and Baldwin, T. (2006). Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 164–171, Stroudsburg, PA, USA. ACL.

Bozsahin, C. (2002). The combinatory morphemic lexicon. *Computational Linguistics*, 28:145–186.

Brent, M. R. (1993). From grammar to lexicon: unsupervised learning of lexical syntax. *Computational Linguistics*, 19(2):243–262.

Brighton, H., Kirby, S., and Smith, K. (2005). Cultural selection for learnability: Three hypotheses concerning the characteristic structure of language. In Tallerman, M., editor, *Language Origins: Perspectives on Evolution*. University Press.

Briscoe, T. and Carroll, J. (1997). Automatic extraction of subcategorization from corpora. In *Proceedings of the fifth conference on Applied natural language processing*, pages 356–363, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Briscoe, T., Carroll, J., and Watson, R. (2006). The second release of the RASP system. In *Proceedings of COLING/ACL '06*, pages 77–80, Morristown, NJ, USA.

Buttery, P. and Korhonen, A. (2007). I will shoot your shopping down and you can shoot all my tins: automatic lexical acquisition from the CHILDES database. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, CACLA '07, pages 33–40, Stroudsburg, PA, USA. ACL.

Cakici, R. (2005). Automatic induction of a CCG grammar for Turkish. In *ACL Student Research Workshop*, pages 73–78. ACL.

Carroll, J., Briscoe, T., and Sanfilippo, A. (1998). Parser evaluation: a survey and a new proposal. In *Proceedings of LREC '98*, pages 447–454.

Carroll, J. and Fang, A. (2004). The automatic acquisition of verb subcategorisations and their impact on the performance of an HPSG parser. In *Proceedings of IJCNLP '04*, pages 107–114, Sanya City, China.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI '97*, pages 598–603.

Cholakov, K. and van Noord, G. (2010a). Acquisition of unknown word paradigms for large-scale grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 153–161, Stroudsburg, PA, USA. ACL.

Cholakov, K. and van Noord, G. (2010b). Using unknown word techniques to learn known words. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 902–912, Stroudsburg, PA, USA. ACL.

Christodoulopoulos, C. (2008). Creating a natural logic inference system with combinatory categorial grammar. Master's thesis, School of Informatics, University of Edinburgh.

Clark, A. (2000). Inducing syntactic categories by context distribution clustering. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning - Volume 7*, pages 91–94, Morristown, NJ, USA. ACL.

Clark, A. (2001). Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 workshop on Computational Natural Language Learning - Volume 7*, ConLL '01, pages 13:1–13:8, Morristown, NJ, USA. ACL.

Clark, S. and Curran, J. R. (2003). Log-linear models for wide-coverage CCG parsing. In *Proceedings of EMNLP '03*, pages 97–104, Morristown, NJ, USA.

Clark, S. and Curran, J. R. (2004). The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of COLING '04*, page 282, Morristown, NJ, USA.

Clark, S. and Curran, J. R. (2006). Partial training for a lexicalized-grammar parser. In *Proceedings of the NAACL-HLT '06*, pages 144–151, Morristown, NJ, USA.

Clark, S. and Curran, J. R. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Clark, S., Hockenmaier, J., and Steedman, M. (2002). Building deep dependency structures with a wide-coverage ccg parser. In *Proceedings of ACL '02*, pages 327–334, Morristown, NJ, USA. ACL.

Clark, S., Steedman, M., and Curran, J. (2004). Object-extraction and question-parsing using CCG. In *Proceedings of EMNLP '04*, pages 111–118.

Cocke, J. (1969). *Programming languages and their compilers: Preliminary notes*. Courant Institute of Mathematical Sciences, New York University.

Cucerzan, S. and Yarowsky, D. (2000). Language independent minimally supervised induction of lexical probabilities. In *Proceedings of ACL 2000*, pages 270–277.

Cussens, J. and Pulman, S. (2000). Incorporating linguistics constraints into inductive logic programming. In *Proceedings of the 2nd workshop on Learning language in*

logic and the 4th conference on Computational natural language learning - Volume 7, ConLL '00, pages 184–193, Stroudsburg, PA, USA. ACL.

Deane, P. (2003). Cooccurrence and constructions. In L. Lagerwerf, W. S. and Desgand, L., editors, *Determination of information and tenor in texts: Multidisciplinary approaches to discourse*, pages 277–304, Amsterdam. Stichting Neerlandistiek & Nodus Publikationen.

Deoskar, T. (2008). Re-estimation of lexical parameters for treebank PCFGs. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 193–200, Morristown, NJ, USA. ACL.

Deoskar, T. (2009). *Induction of Fine-grained Lexical Parameters of Treebank PCFGs with Inside-Ooutside Estimation and Lexical Transformations*. PhD thesis, Cornell University.

Deoskar, T., Mylonakis, M., and Sima'an, K. (2011). Learning structural dependencies of words in the Zipfian tail. In *IWPT*, pages 80–91.

Deoskar, T. and Rooth, M. (2008). Induction of treebank-aligned lexical resources. In *Proceedings of the 6th Language Resources and Evaluation Conference*, pages 3159–3166. ACL.

Deoskar, T., Rooth, M., and Sima'an, K. (2009). Smoothing fine-grained PCFG lexicons. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT '09)*, pages 214–217, Morristown, NJ, USA. ACL.

Doran, C. and Srinivas, B. (1997). Developing a wide-coverage CCG system. Technical report, University of Pennsylvania.

Fouvry, F. (2003). Lexicon acquisition with a large-coverage unification-based grammar. In *EACL*, pages 87–90, Budapest, Hungary.

Graff, D. and Cieri, C. (2003). English gigaword. Technical report, Linguistic Data Consortium, Philadelphia.

Grenager, T. and Manning, C. D. (2006). Unsupervised discovery of a statistical verb lexicon. In *Proceedings of EMNLP '06*, pages 1–8, Sydney, Australia.

Hockenmaier, J. (2003). *Data and models for statistical parsing with Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh, Edinburgh, UK.

Hockenmaier, J. (2006). Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of ACL '06*, pages 505–512, Morristown, NJ, USA. ACL.

Hockenmaier, J. and Steedman, M. (2002). Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of ACL '02*, pages 335–342, Morristown, NJ, USA. ACL.

Hockenmaier, J. and Steedman, M. (2007). CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Hwa, R. (1999). Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th annual meeting of the ACL on Computational Linguistics*, pages 73–79, Morristown, NJ, USA. ACL.

Jaruskulchai, C. (1998). An Automatic Thai Lexical Acquisition from Text. In *PRICAI '98: Proceedings of the 5th Pacific Rim International Conference on Artificial Intelligence*, pages 436–447, London, UK. Springer-Verlag.

Kasami, T. (1965). An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory.

Kato, T. (1994). Yet another chart-based technique for parsing ill-formed input. In *Proceedings of ANLC '94*, pages 107–112, Stroudsburg, PA, USA. ACL.

Kazakov, D., Pulman, S., and Muggleton, S. (1998). The FraCas dataset and the LLL challenge. Technical report, SRI International.

Klein, D. and Manning, C. D. (2005). Natural language grammar induction with a generative constituent-context model. *Pattern Recogn.*, 38(9):1407–1419.

Koo, T., Rush, A., Collins, M., Jaakkola, T., and Sontag, D. (2010). Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA. ACL.

Korhonen, A. (2002). *Subcategorization Acquisition*. PhD thesis, Computer Laboratory, University of Cambridge.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *EMNLP '10*, pages 1223–1233, Stroudsburg, PA, USA. ACL.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *EMNLP '11*, pages 1512–1523, Stroudsburg, PA, USA. ACL.

Levin, B. (1993). *English verb classes and alternations : a preliminary investigation*. University of Chicago Press.

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774, Morristown, NJ, USA.

Manning, C. D. (1993). Automatic acquisition of a large subcategorization dictionary from corpora. In *Meeting of the ACL*, pages 235–242.

Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

McClosky, D., Charniak, E., and Johnson, M. (2006). Effective self-training for parsing. In *Proceedings of NAACL-HLT '06*, pages 152–159.

McClosky, D., Charniak, E., and Johnson, M. (2008). When is self-training effective for parsing? In *Proceedings of COLING '08*, pages 561–568, Morristown, NJ, USA.

McGuffey, W. H. (1836). *McGuffey's First Eclectic Reader*, volume Project Gutenberg Ebook Edition (2005, Don Kostuch). Truman and Smith.

McGuffey, W. H. (1837). *McGuffey's Second Eclectic Reader*, volume Project Gutenberg Ebook Edition (2005, Don Kostuch). Truman and Smith.

Mellish, C. S. (1989). Some chart based techniques for parsing ill-formed input. In *Proceedings of the ACL '89*, pages 102–109, Morristown, NJ, USA. ACL.

Min, K. and Wilson, W. H. (1998a). Integrated control of chart items for error repair. In *Proceedings of the 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 862–868, Stroudsburg, PA, USA. ACL.

Min, K. and Wilson, W. H. (1998b). Integrated control of chart items for error repair. In *Proceedings of the 17th international conference on Computational linguistics - Volume 2*, COLING '98, pages 862–868, Stroudsburg, PA, USA. ACL.

Nigam, K., Mccallum, A. K., Thrun, S., and Mitchell, T. (1999). Text classification from labeled and unlabeled documents using em. In *Machine Learning*, pages 103–134.

Osborne, M. and Briscoe, T. (1997). Learning stochastic categorial grammars. In *Proceedings of CoNLL '97*, pages 80–87. ACL.

Otani, A. and Steedman, M. (2007). Case, Coordination, and Information Structure in Japanese. In *Proceedings of the 21st Pacific Asia Conference on Language, Information and Computation*, pages 365–874.

Preiss, J., Briscoe, T., and Korhonen, A. (2007). A system for large-scale acquisition of verbal, nominal and adjectival subcategorization frames from corpora. In *Proceedings of ACL '07*, pages 912–919, Prague, Czech Republic.

Pullum, G. and Scholz, B. (2007). Systematicity and natural language syntax. *Croatian Journal of Philosophy*, 21:375–402.

Regier, T. and Gahl, S. (2004). Learning the unlearnable: the role of missing evidence. *Cognition*, 93(2):147–155.

Reichart, R. and Rappoport, A. (2007). Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of ACL 2007*.

Rimell, L. and Clark, S. (2008a). Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of EMNLP '08*, pages 475–484, Stroudsburg, PA, USA.

Rimell, L. and Clark, S. (2008b). Constructing a parser evaluation scheme. In *COLING '08: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 44–50, Stroudsburg, PA, USA.

Rimell, L., Clark, S., and Steedman, M. (2009). Unbounded dependency recovery for parser evaluation. In *Proceedings of EMNLP '09*, pages 813–821, Stroudsburg, PA, USA.

Snyder, B. and Barzilay, R. (2010). Climbing the Tower of Babel: Unsupervised multilingual learning. In *ICML*, pages 29–36.

Snyder, B., Naseem, T., and Barzilay, R. (2009). Unsupervised multilingual grammar induction. In *ACL/AFNLP*, pages 73–81.

Steedman, M. (1991). Type-raising and directionality in combinatory grammar. In *Proceedings of the 29th annual meeting on ACL*, ACL '91, pages 71–78, Stroudsburg, PA, USA. ACL.

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

Steedman, M., Baker, S., Crim, J., Clark, S., Hockenmaier, J., Hwa, R., Osborne, M., Ruhlen, P., and Sarkar, A. (2003a). Clsp ws-02 final report: Semi-supervised training for statistical parsing. Technical report, Johns Hopkins University.

Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J., Ruhlen, P., Baker, S., and Crim, J. (2003b). Bootstrapping statistical parsers from small datasets. In *Proceedings of the tenth conference on European chapter of the ACL - Volume 1*, EACL '03, pages 331–338, Morristown, NJ, USA. ACL.

Szolovits, P. (2003). Adding a medical lexicon to an English parser. In *Proceedings of the AMIA 2003 Annual Symposium*.

Thomforde, E. and Steedman, M. (2011). Semi-supervised CCG Lexicon Extension. In *Proceedings of EMNLP '11*, pages 1246–1256, Edinburgh. ACL.

van Noord, G. (2004). Error mining for wide-coverage grammar engineering. In *Proceedings of ACL '04*, pages 446–453, Barcelona, Spain.

Watkinson, S. and Manandhar, S. (1999a). Unsupervised lexical learning of categorial grammars. In *ACL'99: Workshop in Unsupervised Learning in Natural Language Processing*.

Watkinson, S. and Manandhar, S. (1999b). Unsupervised lexical learning with categorial grammars using the LLL corpus. In Cussens, J., editor, *Proceedings of the 1st Workshop on Learning Language in Logic*, pages 16–27, Bled, Slovenia.

Watkinson, S. and Manandhar, S. (2000). Unsupervised lexical learning with categorial grammars using the LLL corpus. In Cussens, J. and Dvzeroski, S., editors, *Learning Language in Logic*, volume 1925 of *Lecture Notes in Artificial Intelligence*. Springer.

Watkinson, S. and Manandhar, S. (2001a). Acquisition of large scale categorial grammar lexicons. In *Proceedings of PACLING '01*.

Watkinson, S. and Manandhar, S. (2001b). A psychologically plausible and computationally effective approach to learning syntax. In Daelemans, W. and Zajac, R., editors, *Proceedings of CoNLL '01)*, pages 160 – 167.

White, M. and Baldridge, J. (2003). Adapting chart realization to CCG. In *Proceedings of the 9th European Workshop on Natural Language Generation*, pages 119–126.

Widdows, D. (2003). Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proceedings of NAACL-HLT '03*, pages 197–204, Morristown, NJ, USA.

Yao, X., Ma, J., Duarte, S., and Coltekin, C. (2009a). An inference-rules based categorial grammar learner for simulating language acquisition. In *Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning*, Tillburg.

Yao, X., Ma, J., Duarte, S., and Coltekin, C. (2009b). Unsupervised syntax learning with categorial grammars using inference rules. In *Proceedings of The 14th Student Session of the European Summer School for Logic, Language, and Information*, Bordeaux.

Yona, S. and Wintner, S. (2008). A finite-state morphological grammar of Hebrew. *Natural Language Engineering*, 14(2):173–190.

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

Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI-05*, pages 658–66, Arlington, Virginia. AUAI Press.

Zhang, Y., Baldwin, T., and Kordoni, V. (2005). The corpus and the lexicon: standardising deep lexical acquisition evaluation. In *Proceedings of the 5th Workshop on Important Unresolved Matters*, pages 152–159. ACL.

Zhang, Y., Baldwin, T., and Kordoni, V. (2007). The corpus and the lexicon: standardising deep lexical acquisition evaluation. In *Proceedings of the Workshop on Deep Linguistic Processing*, DeepLP '07, pages 152–159, Stroudsburg, PA, USA. ACL.

Zhang, Y., Baldwin, T., Kordoni, V., Martinez, D., and Nicholson, J. (2010). Chart mining-based lexical acquisition with precision grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, HLT '10, pages 10–18, Stroudsburg, PA, USA. ACL.