Semi-supervised CCG Lexicon Extension

Emily Thomforde
University of Edinburgh
e.j.thomforde@sms.ed.ac.uk

Mark Steedman
University of Edinburgh
steedman@inf.ed.ac.uk

Abstract

This paper introduces Chart Inference (CI), an algorithm for deriving a CCG category for an unknown word from a partial parse chart. It is shown to be faster and more precise than a baseline brute-force method, and to achieve wider coverage than a rule-based system. In addition, we show the application of CI to a domain adaptation task for question words, which are largely missing in the Penn Treebank. When used in combination with self-training, CI increases the precision of the baseline StatCCG parser over subject-extraction questions by 50%. An error analysis shows that CI contributes to the increase by expanding the number of category types available to the parser, while self-training adjusts the counts.

1 Introduction

Unseen lexical items are a major cause of error in strongly lexicalised parsers such as those based on CCG (Clark and Curran, 2003; Hockenmaier, 2003). The problem is especially acute for less privileged languages, but even in the case of English, we are aware of many category types entirely missing from the Penn Treebank (Clark et al., 2004).

In the case of totally unseen words, the standard method used by StatCCG (Hockenmaier, 2003) and many other treebank parsers is part-of-speech backoff, which is quite effective, affording an F-score of 93% over dependencies in §00 in the optimal configuration. It is difficult to say how backing off affects dependency errors, but when we examine category match accuracy of the CCGBank-trained parser, we find that POS backoff has been used on 19.6% of tokens, which means that those tokens are unseen, or too infrequent in the training data to be included in the lexicon. Of the 3320 items the parser labelled incorrectly, 675 (20.3%) are words that are missing from the lexicon entirely.[1] In the best case, if we were able to learn lexical entries for those 675, we could transfer them to lexical treatment, which is 93.5% accurate, rather than POS backoff, which is 89.3% accurate. Under these conditions, we predict a further 631 word/category pairs to be tagged correctly by the parser, reducing the error rate from 7.4% to 6% on §00. 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.

This paper introduces Chart Inference (CI) as a 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 can be inferred from the types of the surrounding constituents. CI is designed to take advantage of this property of generative CCGBank-trained parser, and of access to the full inventory of CCG combinators and non-combinatory unary rules from the trained model. It is capable of learning category types that are completely missing from the lexicon, and is superior to existing learning systems in both precision and efficiency.

Four experiments are discussed in this paper. The first compares three word-learning methods for their ability to converge to a toy target lexicon. The sec-

[1]A further 269 (8%) are cases where the word is known, but has not been seen with the correct category.

ond and third compare the three methods based on their ability to correctly tag the all the words in a small natural language corpus. The final experiment shows how Chart Induction can be effectively used in a domain adaptation task where a small number of category types are known to be missing from the lexicon.

2 Learning Words

The methods used in this paper all operate under a restricted learning setting, over sentences where all but one word is in the lexicon. Since the learning portion of the algorithm is unsupervised, it has access to an essentially unlimited amount of unlabelled data, and it can afford to skip any sentence that does not conform to the one-unseen-word restriction. Attempting two or more OOL words at a time from one sentence would compound the search space and the error rate. We do not address the much harder problem of hypothesising missing categories for known words, which should presumably be handled by quite other methods, such as prior offline generalization of the lexicon.

2.1 A Brute-force System

One of the early lexical acquisition systems using Categorial Grammar was that of Watkinson and Manandhar (1999; 2000; 2001a; 2001b). 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. However, they do show categorial grammar to be a promising basis for artificial language acquisition, because CCG makes learning the lexicon and learning the grammar the same task (Watkinson and Manandhar, 1999). 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 category types, increased its chances of converging. This approach of automating the learning process differs from the previous language learning methods described, in that it doesn't require the specification of any particular patterns, only knowledge of the grammar formalism.

For this paper, as a baseline, 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 $C_1 ... C_N$ are in the lexicon: $\gamma = arg\max Parse(C_1 ... C_n, \gamma)$. This is equivalent to backing off to the set of all known category types; 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.

2.2 A Rule-based System

Yao et al. (2009a; 2009b) 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 based on a small inventory of inference rules that eliminated ambiguity in the ordering of arguments. 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. 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.

Although Yao et al. (2009b) used the full set of CCG combinators to generate learned categories, they employed a post-processing step to filter spurious categories by checking whether the category

```
DERIVE($[C_1 ... C_n], \beta, \gamma$)

if $\beta = \emptyset$
  then return ($\gamma$)
  else if $C_1 = C_n = X$
  then { $\gamma = \gamma + \beta$;
         DERIVE($X, \emptyset, \gamma$)
  else { if $C_1 \notin S, X$
           then DERIVE($[C_2 ... C_n], \beta \backslash C_1, \gamma$)
         if $C_n \notin S, X$
           then DERIVE($[C_1 ... C_{n-1}], \beta / C_n, \gamma$)
         if $\beta \equiv B$ and $C_1 \equiv B/A$ and $C_1 \notin S, X$
           then DERIVE($[C_2 ... C_n], A, \gamma$)
         if $\beta \equiv B$ and $C_n \equiv B \backslash A$ and $C_n \notin S, X$
           then DERIVE($[C_1 ... C_{n-1}], A, \gamma$)
```

Figure 1: Generalised recursive rule-based algorithm, 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) category set.

participated in a CG-only derivation (using application rules only). This is effective in limiting spurious derivations, but at the expense of reduced recall on those sentences for whose analysis CCG rules of composition etc. are crucial.

Their rules were effective for their toy-scale datasets, but for the purposes of this paper we have implemented a generalised version of the recursive algorithm for use in wide-coverage parsing. This algorithm is outlined in Figure 1. 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.

2.3 Chart Inference

Both Watkinson's and Yao's experiments were fully convergent over toy datasets, 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 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, the rules do not cover the full set of partial parse conditions, and further to that, 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 one-unknown-word-per-sentence learning setting established 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 section shows that it performs as well as the previous two systems on a toy corpus, and the next section proves that it more readily scales to natural language domains.

Mellish (1989) established a two-stage bidirectional chart parser for diagnosing errors in input text. His method 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. His method 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 they did not evaluate 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 or vice-versa.

Where Mellish uses the two-stage parsing process to complete malformed parses, we use it to diagnose unknown lexical items. In addition, we work on the scale of a full grammar and wide-coverage parser, using modern lexical corpora.

Our method is a wrapper for a naive generative CCG parser StatOpenCCG (Christodoulopoulos, 2008), a statistical extension to OpenCCG (White and Baldridge, 2003). In the general case, the parser is trained on all the labelled data available in a particular learning setting, then the learner discovers new lexical items from unlabelled text. Like the brute force and rule-based systems, it is vulnerable

CCG Combinator					**Inverse Combinator**							
A/B	B	$\rightarrow$	A	(>)	$\mathbf{X}$	B	$\rightarrow$	A	$\Rightarrow$	$\mathbf{X} = A/B$	if v(B) ≤ 1
					A/B	$\mathbf{X}$	$\rightarrow$	A	$\Rightarrow$	$\mathbf{X} = B$	
B	$A\backslash B$	$\rightarrow$	A	(<)	$\mathbf{X}$	$A\backslash B$	$\rightarrow$	A	$\Rightarrow$	$\mathbf{X} = B$	
					B	$\mathbf{X}$	$\rightarrow$	A	$\Rightarrow$	$\mathbf{X} = A\backslash B$	if v(B) ≤ 1
A/C	C/B	$\rightarrow$	A/B	(>B)	$\mathbf{X}$	C/B	$\rightarrow$	A/B	$\Rightarrow$	$\mathbf{X} = A/C$	
					A/C	$\mathbf{X}$	$\rightarrow$	A/B	$\Rightarrow$	$\mathbf{X} = C/B$	
$C\backslash B$	$A\backslash C$	$\rightarrow$	$A\backslash B$	(<B)	$\mathbf{X}$	$A\backslash C$	$\rightarrow$	$A\backslash B$	$\Rightarrow$	$\mathbf{X} = C\backslash B$	
					$C\backslash B$	$\mathbf{X}$	$\rightarrow$	$A\backslash B$	$\Rightarrow$	$\mathbf{X} = A\backslash C$	

Figure 2: Derivation of inverse combinators

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

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

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

Figure 3: CI probability that the target is category C, given possible categories for result (R) and sister (S).

to attachment errors and ambiguity from adverbials.

The learning step consists in presenting the parser with sentences all of whose words but one are in-lexicon. The parser must have a statistical parsing model, which contains a seed lexicon, a set of CCG combinators, and an optional set of unary and binary rules learned from the training corpus.

First the baseline bottom-up parser is called upon to produce a partial parse chart. The learner takes this partial chart and fills the top right cell with a distribution for the result category based on the end punctuation.[2]

Using this partial chart that contains at least one entry for every leaf cell (except the one OOL target cell) and at least one entry for the result, the learner steps through the chart in a top-down version of CYK (Younger, 1967). For the top-down process, the standard 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. This process continues until the target cell has been filled, and the ranked set of categories is returned.

The probability that the target has a given category is calculated as the greater of the right- or left-headed derivations, according to Figure 3. 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. Where a value is unavailable, it backs off to a pre-specified value (default 0.0001).[3] The system requires a pruning parameter that limits each cell to the top N most probable categories. Here, we set N=10, to limit the search space and complexity. [4]

Figure 2 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. In the two permissive instances where the sister category's form is the unrestricted *B*, we limit the sister's valency to 1, in order to keep the learner from generating spurious categories that could result from these two rules being overapplied.

Figure 4 illustrates the workings of the learning

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

[3]This backoff parameter allows adjustment of the expectation of new category types and could be replaced with another smoothing method in subsequent implementations.

[4]Further testing on the McGuffey corpus has shown the average rank of correct tags in the category set to be 1.4.

algorithm for the sentence *The cat **X** her*. The grey cells are filled as a partial chart by the parser, and the white cells are filled by the top-down learner. Note that taking rule probabilities into account makes the algorithm robust to ambiguity. The highest-ranking lexical category for *her* is $NP[nb]/N$, but the next highest (*NP*) is preferred in the derivation of the highest-ranking category for the unknown word **X**.

3 Experiment I: Convergence

In the following experiments, we compare Chart Inference to the two baseline methods: Brute Force (BF), derived from Watkinson and Manandhar, and Rule-Based (RB), derived from Yao et al. This section investigates how robust the three systems are to changes in theoriginal seed lexicon.

3.1 Corpus

For this experiment we test the three systems on a reconstructed version of Corpus 1 from Watkinson and Manandhar's experiments.[5] The lexicon 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*). The test sentences are randomly generated from a simple PCFG over the lexicon, and are always presented to the learners in the same order.

3.2 Methods

In order to directly compare the three learning methods, we use the evaluation setting from Watkinson and Manandhar (1999), which consists of a 40-entry target lexicon and a PCFG language model used to randomly generate 1000 sentences. We then specify a seed lexicon and run the learner incrementally, so that it deals with one sentence at a time, then feeds the learned material back into the lexicon. Watkinson's system was shown to fully convergent (they defined convergence as cosine similarity between the seed lexicon ($\vec{S}$) and the target lexicon ($\vec{T}$) exceeding 0.99), whenever the seed lexicon contained at least one instance of each of the category types in the target lexicon (Watkinson and Manandhar, 1999)

[5]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 from the original in the distribution of category types. The reconstructed corpus will be released shortly.

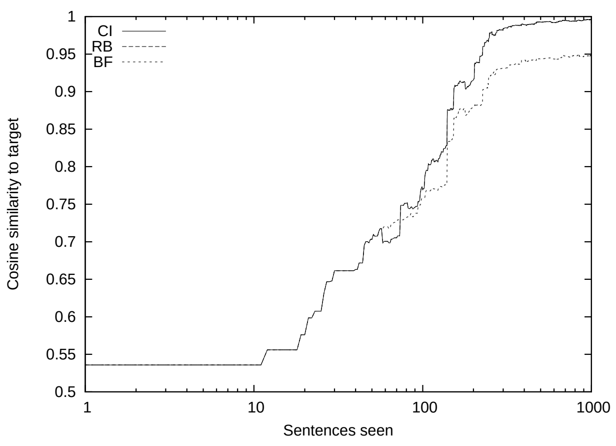

Figure 5: Learning curve for all three methods when the seed contains no ditransitives. CI and RB are identical.

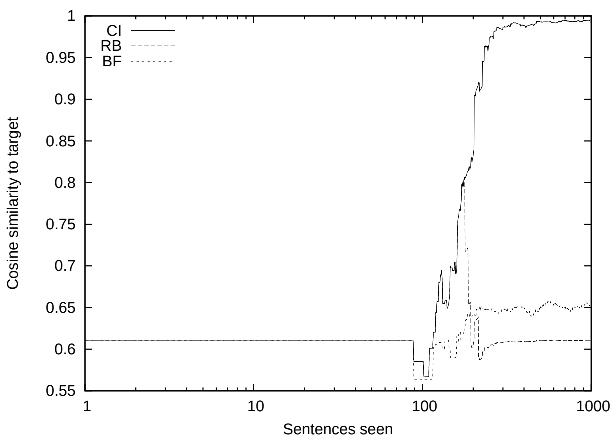

Figure 6: Learning curve for all three methods when seed contains only three determiners and one noun.

3.3 Results

When run incrementally over this toy corpus, both the RB and CI algorithms converge to the target lexicon in an identical sigmoid learning curve (not shown). However, when we start with an impoverished seed, the algorithms' behaviours start to diverge. Figure 5 shows the learning curve for the three methods when the seed lexicon omits all instances 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 plateaus at 95% similarity.

When the seed is reduced to only three determiners and a noun, CI can still learn the complete

0	1	2	3	
$NP[nb]/N$: 0.08428	$NP[nb]$: 0.00138	$S[dcl]\backslash NP$: 2.90E-5 $S[dcl]/NP$: 2.90E-10	$S[dcl]$: 1.0	0
The	N : 0.01890 NP : 0.00174 $S/(S\backslash NP)$: 0.00152	$(S[dcl]\backslash NP)/NP$: 1.35E-7	$S[dcl]\backslash NP$: 1.00E-4	1
	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 ...	2
		X	$NP[nb]/N$: 0.05467 NP : 0.02439 $S/(S\backslash NP)$: 0.02124	3
			her	.

Figure 4: Example of a two-stage derivation using Chart Inference: Grey boxes are filled bottom-up by the partial parser; white boxes top-down by the learner. The target cell (2,2) shows the correct category type as the highest probability solution.

lexicon, despite some initial missteps and a steeper curve. However, the other two methods fail catastrophically (Figure 6). BF never gets going, since it can only correctly learn the remaining nouns. RB is partially successful, but is thwarted by a bad decision at 80% 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.

4 Experiments II and III: Coverage

Next, we compare the three learning methods on a larger corpus of natural language, to investigate how well they perform at recovering a wide range of category types in complex settings.

4.1 Corpus

We have constructed a small natural language lexicon based on the first volume of a 6-volume 1836 children's primer, McGuffey's Eclectic Reader.[6] Volume 1 of the McGuffey corpus (MG1) consists of 546 sentences that have been manually annotated with CCG categories, automatically parsed, and then corrected. Volume 2 (MG2) comprises 801 sentences, annotated in the same manner as Volume 1, though not as reliably. The McGuffey corpus makes an ideal seed for development purposes, as it contains a high proportion of simple declarative sentences, but also touches on questions, quotations, passives, and other complex constructions.

[6]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 will be released shortly.

4.2 Methods

In the first of these two experiments we train and test on the same 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 addresses the lexical coverage problem discussed in Section 1 of this paper.

The second of these two experiments looks at a more realistic environment for word learning: the parser is initially trained on MG1, then tested on MG2. We evaluate on the gold standard categories in MG2. 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 will inevitably be lower than the first.

Figure 7 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.

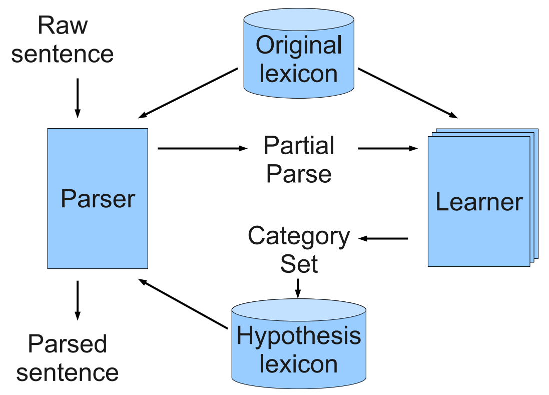

Figure 7: Learning framework for Experiments II-IV.

4.3 Results

Table 1 compares the category match accuracy across the three systems in experiment II, as well as the baseline that chose the the most probable category for the target word's POS. Two tasks are scored: *Top One*, where we evaluate the single highest-scoring category against the gold-standard tag, and *Top Ten*, where we check to see if the gold tag is in the set of the ten highest-probability categories returned by the learner.

CI achieves the best F-scores in both tasks, reaching 76% for *Top One* and 94% for *Top Ten*. POS backoff has an advantage in the *Top Ten* task, especially in recall, since it returns an answer in every case, but CI still outperforms it on F-score. BF achieves the highest precision in the *Top One* task, but takes 30 hours to do so, since it is searching over all possible categories. RB is markedly worse in both precision and recall, 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.

In Experiment III, to test the limits of the learners on truly OOL words, we again train on MG1, but test instead on MG2. 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.

Out of its 801 sentences in MG2, only 32 present learning opportunities for the learners, being between 2 and 10 tokens long, containing no internal punctuation or coordination, and containing only one OOL word.

Table 2 shows the category match results of the three systems on MG2. Recall is calculated over the set of learning opportunities, of which there are only 32. BF performs best in all metrics, but the CI results are reasonable. The underlying reason for this behaviour is that the 32 learning targets are all of common categories: over half of them are N or NP. Since the Brute Force learner seeks simply to maximise the tree probability, N and NP are its most common guesses in general.

5 Experiment IV: Domain Adaptation

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 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 Chart Inference to learn WH-question words from the *unlabelled* question corpus. If successful, this would eliminate the human-annotation step for domain adaptation of the kind investigated by (Clark et al., 2004).

5.1 Corpora

We trained the initial parser on the CCG-Bank (Hockenmaier and Steedman, 2007; Hockenmaier, 2003) training set (§02-21), consisting of 39603 sentences of Wall Street Journal text (Marcus et al., 1993). It is important to note that this training corpus contains only 93 questions in total, so it is not surprising that several category types for question words are entirely unrepresented. It also rein-

	Top One			Top Ten			
	P	R	F	P	R	F	Time (m)
POS	64.91	64.91	64.91	92.55	**92.55**	92.55	1
BF	**80.53**	65.84	72.45	95.97	78.32	86.25	1740
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 1: Exp. II: Category match results for the three systems on the McGuffey corpus, training and testing on MG1.

	Top One			Top Ten		
	P	R	F	P	R	F
BF	70.83	53.13	60.72	83.33	62.50	71.43
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 2: Exp. III: Category match results for the three systems on the McGuffey corpus, training on MG1 and testing on MG2. Common categories (N, NP, N/N) are overrepresented in the test data, leading to higher BF scores. POS tags not available for MG2, so no POS baseline is reported.

forces the fact that this is a domain-adaptation task.

We use the same 500-sentence test set as Rimell and Clark (2008b). The test corpus consists of 488 questions, each starting with *What*, *When*, *How*, *Who* or *Where*. The learning corpus contains 1328 questions in a similar distribution.

Only three out of the five categories needed to parse What-questions are present in the CCGBank seed lexicon: $S[wq]/(S[q]/NP)$,[7], $S[wq]/(S[dcl]\backslash NP)$,[8] and $S[wq]/(S[q]/NP)/N$.[9] For this experiment we focus on the subject WH-element extraction category $(S[wq]/(S[dcl]\backslash NP))/N$, as in *Which cat is the grandmother?*. This particular category was chosen as a point of investigation because it is OOL in CCGBank and is common enough to meaningfully evaluate.

5.2 Methods

The baseline is the original StatCCG parser and lexicon. We also employ self-training (Charniak, 1997), in which a parser is used to parse a set of sentences, and then retrained using those output trees. Self-training has had very little success in CCG applications hitherto. McClosky et al (2006) attribute success in self-training 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). We follow their recommendations that the best performance is achieved when all the training sentences are parsed at once, rather than incrementally.

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 Rimmell and Clark (2008a). Second, we add the parsed questions into the training set, then retrain and finally retest the parser.

The parser was initially trained on CCGBank §02-21 with a word frequency threshold of 5.[10] 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.

We run the learner on the set of 149 sentences from the TREC Question-Answering corpus (Rimell and Clark, 2008b) that contain the word/category pair *What*:$(S[wq]/(S[dcl]\backslash NP))/N$. For this experiment the end-punctuation distribution derived from the training corpus is replaced with a single value: $P(S[wq]|?) = 1$.

[7]Object question category as in *What is the Keystone State?*

[8]Subject question category as in *What lays blue eggs?*

[9]Object WH-element extraction category as in *What continent is Scotland in?*

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

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

Table 3: F-score over individual category matches. Bold means significantly different from the Baseline.

5.3 Results

Table 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 subject WH-element-extraction 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 Chart Inference 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. [11] The reason for this can be clearly seen when we evaluate the lexicons created by each method.

Table 4 shows the differences in the impact on the lexicon between baseline (BL), Chart Induction (CI), and the combined method of CI and self-training (CI+ST).[12] 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, 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, as we saw in Table 3.

[11] 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.

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

5.4 Error Analysis

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 are classified in Table 4 as either rare (R), spurious (*), or duplicate (D). 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 duplicate 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. In essence, it cannot see the difference between *What companies have them?* and *What choice have they?* when the NPs lack a case distinction. 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.

As seen in Table 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. Duplicate categories can be considered spurious for the sake of parsing, but

		$P(W\|C)$			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
D	$(S[wq]/S[inv])/N$	-	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 4: Exp. IV: 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 duplicate (D). Cateogories above the middle line are present in the Baseline lexicon; below are induced.

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.

6 Conclusion and Future Work

Chart Inference is a useful tool for finding OOL categories. It has been shown to outperform both the brute-force and rule-based systems. When used in conjunction with self-training, CI presents a valuable framework for domain adaptation in the case where whole category types are missing from the lexicon.

It remains to put Chart Inference into an appropriate framework for improving coverage over the baseline WSJ-trained StatCCG parser. We estimate an upper bound of 20% error reduction possible over CCGBank §00, if the lexicon is expanded to cover all the necessary word/category pairs. Improving global F-score for §23 is of course very difficult. The lexical entries CI finds are by definition rare and at the scale we are running, they are unlikely to occur in those 2000 sentences. We believe our analysis of the lexical items themselves shows that we are learning a high proportion of good lexical entries.

The problem of discovering missing categories for known words remains. We have shown through adapting to the question domain that it is possible to make focused improvements when we can identify the gaps in coverage (as in wh-question words), but in order to address the challenge of automatic lexicon extension fully, quite different techniques for generalising lexical entries for seen words will be require.

Acknowledgements

The authors would like to thank Steve Clark and Laura Rimell for provision of the annotated QA corpus, Christos Christodoulopoulos for the StatOpenCCG parser, Tejaswini Deoskar for editorial advice, and Luke Zettlemoyer for considerable mathematical assistance. This work was partially funded by EU ERC Advanced Fellowship 249520 GRAMPLUS, IST Cognitive Systems IP EC-FP7-270273 "XPERIENCE" and a grant from Wolfson Microelectronics.

References

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI '97*, pages 598–603.

Christos Christodoulopoulos. 2008. Creating a natural logic inference system with combinatory categorial grammar. Master's thesis, School of Informatics, University of Edinburgh.

Stephen Clark and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of EMNLP '03*, pages 97–104, Morristown, NJ, USA.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark, Mark Steedman, and James Curran. 2004. Object-extraction and question-parsing using CCG. In *Proceedings of EMNLP '04*, pages 111–118.

Julia Hockenmaier and Mark Steedman. 2007. CCG-bank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Julia Hockenmaier. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

Tsuneaki Kato. 1994. Yet another chart-based technique for parsing ill-formed input. In *Proceedings of ANLC '94*, pages 107–112, Stroudsburg, PA, USA. Association for Computational Linguistics.

D. Kazakov, S. Pulman, and S. Muggleton. 1998. The FraCas dataset and the LLL challenge. Technical report, SRI International.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330, June.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of NAACL-HLT '06*, pages 152–159.

David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *Proceedings of COLING '08*, pages 561–568, Morristown, NJ, USA.

Chris S. Mellish. 1989. Some chart based techniques for parsing ill-formed input. In *Proceedings of the ACL '89*, pages 102–109, Morristown, NJ, USA. Association for Computational Linguistics.

Laura Rimell and Stephen Clark. 2008a. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of EMNLP '08*, pages 475–484, Stroudsburg, PA, USA.

Laura Rimell and Stephen Clark. 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.

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

Stephen Watkinson and Suresh Manandhar. 1999. Unsupervised lexical learning of categorial grammars. In *ACL'99: Workshop in Unsupervised Learning in Natural Language Processing*.

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

Stephen Watkinson and Suresh Manandhar. 2001a. Acquisition of large scale categorial grammar lexicons. In *Proceedings of PACLING '01*.

Stephen Watkinson and Suresh Manandhar. 2001b. A psychologically plausible and computationally effective approach to learning syntax. In Walter Daelemans and R'emi Zajac, editors, *Proceedings of CoNLL '01)*, pages 160 – 167.

Michael White and Jason Baldridge. 2003. Adapting chart realization to CCG. In *Proceedings of the 9th European Workshop on Natural Language Generation*, pages 119–126.

Xuchen Yao, Jianqiang Ma, Sergio Duarte, and Cagri Coltekin. 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.

Xuchen Yao, Jianqiang Ma, Sergio Duarte, and Cagri Coltekin. 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.

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