

# Object-Extraction and Question-Parsing using CCG

**Stephen Clark and Mark Steedman**

School of Informatics  
University of Edinburgh  
2 Buccleuch Place, Edinburgh, UK  
{stevec, steedman}@inf.ed.ac.uk

**James R. Curran**

School of Information Technologies  
University of Sydney  
NSW 2006, Australia  
james@it.usyd.edu.au

## Abstract

Accurate dependency recovery has recently been reported for a number of wide-coverage statistical parsers using Combinatory Categorical Grammar (CCG). However, overall figures give no indication of a parser’s performance on specific constructions, nor how suitable a parser is for specific applications. In this paper we give a detailed evaluation of a CCG parser on object extraction dependencies found in WSJ text. We also show how the parser can be used to parse questions for Question Answering. The accuracy of the original parser on questions is very poor, and we propose a novel technique for porting the parser to a new domain, by creating new labelled data at the lexical category level only. Using a supertagger to assign categories to words, trained on the new data, leads to a dramatic increase in question parsing accuracy.

## 1 Introduction

Several wide-coverage statistical parsers have recently been developed for Combinatory Categorical Grammar (CCG; Steedman, 2000) and applied to the WSJ Penn Treebank (Clark et al., 2002; Hockenmaier and Steedman, 2002; Hockenmaier, 2003b; Clark and Curran, 2004b). One motivation for using CCG is the recovery of the long-range dependencies inherent in phenomena such as coordination and extraction. Recovery of these dependencies is important for NLP tasks which require semantic interpretation and for processing text which contains a high frequency of such cases, e.g. *Wh*-questions fed to a Question Answering (QA) system.

One shortcoming of treebank parsers such as Collins (1999) and Charniak (2000) is that they typically produce phrase-structure trees containing only local syntactic information. Johnson (2002) uses post-processing methods to insert “empty” nodes into the trees, and Dienes and Dubey (2003) use pre-processing methods to determine where discontinuities are likely to appear in the sentence. In contrast, the CCG parsers detect long-range dependencies as an integral part of the parsing process.

The CCG parser used here (Clark and Curran, 2004b) is highly accurate and efficient, recovering labelled dependencies with an overall F-score of over 84% on WSJ text, and parsing up to 50 sentences per second. Thus the parser should be useful for large-scale NLP tasks. However, the overall accuracy figures give no indication of the parser’s performance on specific constructions, nor how suitable the parser is for specific applications. In this paper we give a detailed evaluation for object extraction dependencies and show how the parser can be used to parse questions for QA.

We find that the parser performs well on the object extraction cases found in the Penn Treebank, given the difficulty of the task. In contrast, the parser performs poorly on questions from TREC, due to the small number of questions in the Penn Treebank. This motivates the remainder of the paper, in which we describe the creation of new training data consisting of labelled questions. Crucially, the questions are labelled at the lexical category level only, and not at the derivation level, making the creation of new labelled data relatively easy.

The parser uses a supertagger to assign lexical categories to words, and the supertagger can be adapted to the new question domain by training on the newly created data. We find that using the original parsing model with the new supertagger model dramatically increases parsing accuracy on TREC questions, producing a parser suitable for use in a QA system. For evaluation we focus on *What* questions used in the TREC competitions. As well as giving an overall evaluation on this test set, we also consider a number of object extraction cases.

The creation of new training data at the lexical category level alone is a technique which could be used to rapidly port the parser to other domains. This technique may also be applicable to other lexicalised grammar formalisms, such as Tree Adjoining Grammar (Bangalore and Joshi, 1999).<sup>1</sup>

<sup>1</sup>Doran et al. (1997) propose using a supertagger for semi-automatically porting the XTAG grammar to a new domain.

## 2 The Parser

The parser used in this paper is described in Clark and Curran (2004b). It takes as input a POS tagged sentence with a set of lexical categories assigned to each word. The CCG combinatory rules are used to combine the categories. A packed chart efficiently represents all of the possible analyses for a sentence, and the CKY chart parsing algorithm described in Steedman (2000) is used to build the chart.

A Maximum Entropy CCG supertagger (Clark and Curran, 2004a) is used to assign the categories. The lexical category set is obtained from CCGbank (Hockenmaier, 2003a), a treebank of normal-form CCG derivations derived from the Penn Treebank. CCGbank is also used for learning the parameters of the supertagger and parsing models.

### 2.1 The Supertagger

The supertagger uses a log-linear model to define a distribution for each word over the lexical category set. Model features are defined by the words and POS tags in the 5-word window surrounding the target word. The supertagger selects the most probable categories locally rather than maximising the sequence probability, assigning all categories whose probability is within some factor,  $\beta$ , of the highest probability category. For a word seen frequently in the training data, the supertagger can only assign categories from the word’s entry in the *tag dictionary*, which lists the categories each word has been seen with in the data.

In Clark et al.’s (2002) parser, a supertagger is used as follows: first around 4 lexical categories are assigned to each word, on average; if the chart gets too big or parsing takes too long, the number of categories is reduced until the sentence can be parsed.

In this paper we use our more recent approach (Clark and Curran, 2004a): first a small number of categories is assigned to each word, e.g. 1.5, and the parser requests more categories if a spanning analysis cannot be found. This method relies on the grammar being constraining enough to decide whether the categories provided by the supertagger are likely to contain the correct sequence. Section 6 shows that this approach works well for parsing questions.

### 2.2 Parsing Model

In Clark and Curran (2004b) we investigate several log-linear parsing models for CCG. In this paper we use the following conditional model:

$$p(y|x) = \frac{1}{Z(x)} e^{\sum_i \lambda_i f_i(y)} \quad (1)$$

where  $y$  is a normal-form derivation and  $x$  is a sentence. (A normal-form derivation is one where com-

position and type-raising are used only when necessary.) There are various features,  $f_i$ , used by the model: rule instantiation features which count the number of times a local tree occurs in a derivation; features defined by the root category of a derivation; and features defined by the lexical categories at the leaves. Each feature type has unlexicalised and head-lexicalised versions.

The remaining features capture word-word dependencies, which significantly improve accuracy. The best-performing model encodes word-word dependencies in terms of the local rule instantiations, as in Hockenmaier and Steedman (2002). We have also tried predicate-argument dependencies, including long-range dependencies, but these have not improved performance. Note we still *recover* long-range dependencies, even if modelling them does not improve performance.

The parser returns a derived structure corresponding to the most probable derivation. For evaluation the parser returns dependency structures, but we have also developed a module which builds first-order semantic representations from the derivations, which can be used for inference (Bos et al., 2004).

## 3 Object Extraction

Steedman (1996) presents a detailed study of various extraction phenomena. Here we focus on object extraction, since the dependencies in such cases are unbounded, and CCG has been designed to handle these cases. Correct dependency recovery for object extraction is also difficult for shallow methods such as Johnson (2002) and Dienes and Dubey (2003).

We consider three types of object extraction: object relative clauses, free object relatives, and *tough*-adjectives (Hockenmaier, 2003a). Examples of the first two from CCGbank are given in Figures 1 and 2, together with the normal-form derivation. The caption gives the number of sentences containing such a case in Sections 2-21 of CCGbank (the training data) and Section 00 (development data).

The pattern of the two derivations is similar: the subject of the verb phrase missing an object is type-raised (**T**); the type-raised subject composes (**B**) with the verb-phrase; and the category for the relative pronoun ( $((NP \setminus NP)/(S[dcl]/NP)$  or  $NP/(S[dcl]/NP)$ ) applies to the sentence-missing-its-object ( $S[dcl]/NP$ ). Clark et al. (2002) show how the dependency between the verb and object can be captured by co-indexing the heads of the *NPs* in the relative pronoun category.

Figure 3 gives the derivation for a *tough*-adjective. The dependency between *take* and *That* can be recovered by co-indexing the heads of *NPs* in

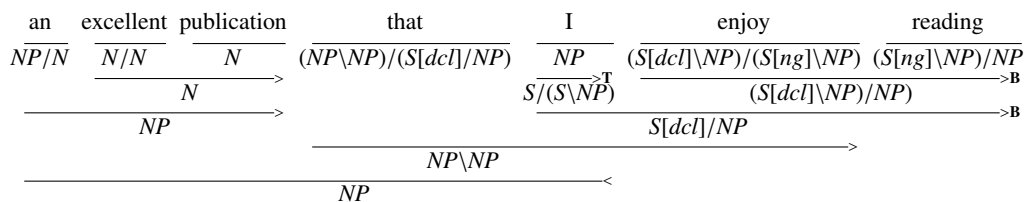


Figure 1: Extraction from object relative clause; 431 sentences in Sections 2-21, 20 in Section 00

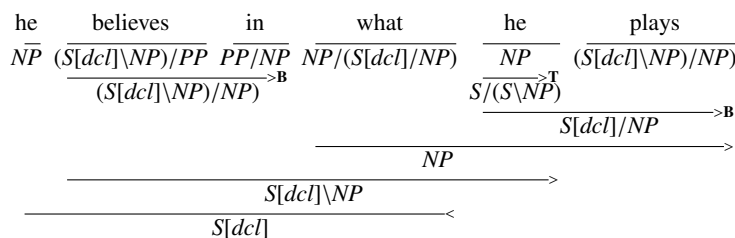


Figure 2: Free object relative example; 269 sentences in Sections 2-21, 16 sentences in Section 00

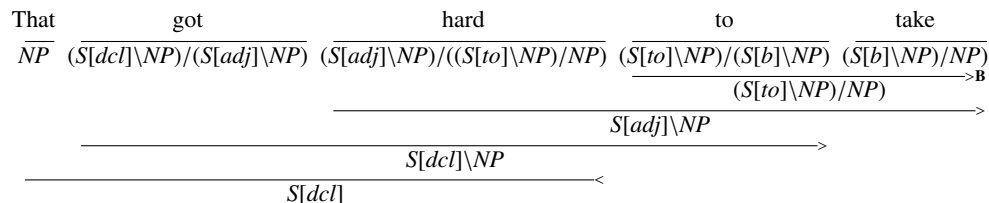


Figure 3: *tough*-adjective example; 52 sentences in Sections 2-21, 2 sentences in Section 00

the categories for *hard* and *got*. These cases are relatively rare, with around 50 occurring in the whole of the treebank, and only two in the development set; the parser correctly recovers one of the two object dependencies for the *tough*-adjective cases in 00.

For the free object relative cases in Section 00, the parser recovers 14 of the 17 gold-standard dependencies<sup>2</sup> between the relative pronoun and the head of the relative clause. The precision is 14/15. For the three gold standard cases that are misanalysed, the category  $NP/S[dcI]$  is assigned to the relative pronoun, rather than  $NP/(S[dcI]/NP)$ .

For the cases involving object relative clauses the parser provides a range of errors for which it is useful to give a detailed analysis.

### 3.1 Analysis of Object Extraction Cases

Figure 4 gives the 20 sentences in Section 00 which contain a relative pronoun with the category  $(NP\NP)/(S[dcI]/NP)$ . There are 24 object dependencies in total, since some sentences contain more than one extraction (11), and some extractions involve more than one head (8, 18, 19). For evaluation, we determined whether the parser correctly re-

covered the dependency between the head of the extracted object and the verb. For example, to get the two dependencies in sentence 18 correct, the parser would have to assign the correct lexical category to *had*, and return *respect* and *confidence* as objects.

The parser correctly recovers 15 of the 24 object dependencies.<sup>3</sup> Overall the parser hypothesises 20 extracted object dependencies, giving a precision of 15/20. Hockenmaier (2003a) reports similar results for a CCG parser using a generative model: 14/24 recall and 14/21 precision. The results here are a significant improvement over those in Clark et al. (2002), in which only 10 of the 24 dependencies were recovered correctly. Below is a detailed analysis of the mistakes made by the parser.

For Sentence 1 the parser cannot provide any analysis. This is because the correct category for *estimated*,  $((S[pt]\NP)/PP)/NP$ , is not in the tag dictionary's entry for *estimated*. Since *estimated* occurs around 200 times in the data, the supertagger only considers categories from the tag dictionary entry, and thus cannot provide the correct category as an option.

<sup>2</sup>One of the 16 sentences contains two such dependencies.

<sup>3</sup>Unless stated otherwise the parser uses automatically assigned, rather than gold standard, POS tags.

1. Commonwealth Edison now faces an additional court-ordered *refund* on its summer/winter rate differential collections *that* the Illinois Appellate Court has *estimated* at \$140 million.
2. Mrs. Hills said many of the 25 *countries that she placed* under varying degrees of scrutiny have made genuine progress on this touchy issue.
- √ 3. It's the petulant complaint of an impudent *American whom Sony hosted* for a year while he was on a Luce Fellowship in Tokyo – to the regret of both parties.
- √ 4. It said the *man, whom it did not name*, had been found to have the disease after hospital tests.
5. Democratic Lt. Gov. Douglas Wilder opened his gubernatorial battle with Republican Marshall Coleman with an abortion *commercial* produced by Frank Greer *that* analysts of every political persuasion *agree* was a tour de force.
6. Against a shot of Monticello superimposed on an American flag, an announcer talks about the strong *tradition* of freedom and individual liberty *that* Virginians have *nurtured* for generations.
- √ 7. Interviews with analysts and business people in the U.S. suggest that Japanese capital may produce the economic *cooperation that* Southeast Asian politicians have *pursued* in fits and starts for decades.
8. Another was Nancy Yeargin, who came to Greenville in 1985, full of the *energy and ambitions that* reformers wanted to *reward*.
9. Mostly, she says, she wanted to prevent the *damage* to self-esteem *that* her low-ability students would *suffer* from doing badly on the test.
- √ 10. Mrs. Ward says that when the cheating was discovered, she wanted to avoid the morale-damaging public *disclosure that* a trial would *bring*.
- √ 11. In CAT sections where students' knowledge of two-letter consonant sounds is tested, the authors noted that Scoring High concentrated on the same *sounds that* the test *does* – to the exclusion of other *sounds that* fifth graders should *know*.
- √ 12. Interpublic Group said its television programming *operations – which it expanded* earlier this year – agreed to supply more than 4,000 hours of original programming across Europe in 1990.
13. Interpublic is providing the programming in return for advertising *time, which it said* will be valued at more than \$75 million in 1990 and \$150 million in 1991.
- √ 14. Mr. Sherwood speculated that the *leeway that* Sea Containers *has* means that Temple would have to substantially increase their bid if they're going to top us.
- √ 15. The Japanese companies bankroll many small U.S. companies with promising products or ideas, frequently putting their money behind *projects that* commercial banks won't *touch*.
- √ 16. In investing on the basis of future transactions, a role often performed by merchant banks, trading companies can cut through the *logjam that* small-company owners often *face* with their local commercial banks.
17. A high-balance *customer that* banks *pine for*, she didn't give much thought to the rates she was receiving, nor to the fees she was paying.
- √ 18. The events of April through June damaged the *respect and confidence which* most Americans previously *had* for the leaders of China.
- √ 19. He described the situation as an escrow *problem, a timing issue, which he said* was rapidly rectified, with no losses to customers.
- √ 20. But Rep. Marge Roukema (R., N.J.) instead praised the House's acceptance of a new youth training wage, a *subminimum that* GOP administrations have *sought* for many years.

Figure 4: Cases of object extraction from a relative clause in 00; the extracted object, relative pronoun and verb are in italics; for sentences marked with a √ the parser correctly recovers all dependencies involved in the object extraction.

For Sentence 2 the correct category is assigned to the relative pronoun *that*, but a wrong attachment results in *many* as the object of *placed* rather than *countries*.

In Sentence 5 the incorrect lexical category  $((S\backslash NP)\backslash(S\backslash NP))/S[dc]$  is assigned to the relative pronoun *that*. In fact, the correct category is provided as an option by the supertagger, but the parser is unable to select it. This is because the category for *agree* is incorrect, since again the correct category,  $((S[dc]\backslash NP)\backslash NP)/(S[dc]\backslash NP)$ , is not in the verb's entry in the tag dictionary.

In Sentence 6 the correct category is assigned to the relative pronoun, but a number of mistakes elsewhere result in the wrong noun attachment.

In Sentences 8 and 9 the complementizer category  $S[em]/S[dc]$  is incorrectly assigned to the relative pronoun *that*. For Sentence 8 the correct analysis is available but the parsing model chose incorrectly. For Sentence 9 the correct analysis is unavailable because the correct category for *suffer*,  $((S[b]\backslash NP)\backslash PP)\backslash NP$ , is not in the verb's entry in the tag dictionary.

In Sentence 13 the correct category is again assigned to the relative pronoun, but a wrong attachment results in *return* being the object of *placed*,

rather than *time*.

In Sentence 17 the wrong category  $S[em]/S[b]$  is assigned to the relative pronoun *that*. Again the problem is with the category for the verb, but for a different reason: the POS tagger incorrectly tags *pine* as a base form (VB), rather than VBP, which completely misleads the supertagger.

This small study only provides anecdotal evidence for the reasons the parser is unable to recover some long-range object dependencies. However, the analysis suggests that the parser fails largely for the same reasons it fails on other WSJ sentences: wrong attachment decisions are being made; the lexical coverage of the supertagger is lacking for some verbs; the model is sometimes biased towards incorrect lexical categories; and the supertagger is occasionally led astray by incorrect POS tags.

Note that the recovery of these dependencies is a difficult problem, since the parser must assign the correct categories to the relative pronoun and verb, and make two attachment decisions: one attaching the relative pronoun to the verb, and one attaching it to the noun phrase. The recall figures for the individual dependencies in the relative pronoun category are 16/21 for the verb attachment and 15/24 for the noun attachment.

In conclusion, the kinds of errors made by the parser suggest that general improvements in the coverage of the lexicon and parsing models based on CCGbank will lead to better recovery of long-range object dependencies.

#### 4 Parsing Questions

Wide-coverage parsers are now being successfully used as part of open-domain QA systems, e.g. Pasca and Harabagiu (2001). The speed and accuracy of our CCG parser suggests that it could be used to parse answer candidates, and we are currently integrating the parser into a QA system. We would also like to apply the parser to the questions, for two reasons: the use of CCG allows the parser to deal with extraction cases, which occur relatively frequently in questions; and the comparison of potential answers with the question, performed by the answer extraction component, is simplified if the same parser is used for both.

Initially we tried some experiments applying the parser to questions from previous TREC competitions. The results were extremely poor, largely because the questions contain constructions which appear very infrequently, if at all, in CCGbank.<sup>4</sup> For example, there are no *What* questions with the general form of *What President became Chief Justice after his presidency?* in CCGbank, but this is a very common form of *Wh*-question. (There is a very small number (3) of similar question types beginning *How* or *Which* in Sections 2–21.)

One solution is to create new annotated question data and retrain the parser, perhaps combining the data with CCGbank. However, the creation of gold-standard derivation trees is very expensive.

A novel alternative, which we pursue here, is to annotate questions at the lexical category level only. Annotating sentences with lexical categories is simpler than annotating with derivations, and can be done with the tools and resources we have available. The key question is whether training only the supertagger on new question data is enough to give high parsing accuracy; in Section 6 we show that it is. The next Section describes the creation of the question corpus.

#### 5 A *What*-Question Corpus

We have created a corpus consisting of 1,171 questions beginning with the word *What*, taken from the TREC 9–12 competitions (2000–2003). We chose to focus on *What*-questions because these are a com-

1. What are Cushman and Wakefield known for?
2. What are pomegranates?
3. What is hybridization?
4. What is Martin Luther King Jr.'s real birthday?
5. What is one of the cities that the University of Minnesota is located in?
6. What do penguins eat?
7. What amount of folic acid should an expectant mother take daily?
8. What city did the Flintstones live in?
9. What instrument is Ray Charles best known for playing?
10. What state does Martha Stewart live in?
11. What kind of a sports team is the Wisconsin Badgers?
12. What English word contains the most letters?
13. What king signed the Magna Carta?
14. What caused the Lynmouth floods?

Figure 5: Examples from the *What*-question corpus

CATEGORY FOR <i>What</i>	FREQ	%
$S[wq]/(S[q]/NP)$	728	62.2
$(S[wq]/(S[q]/NP))/N$	221	18.9
$(S[wq]/(S[dcI]\NP))/N$	207	17.7
$S[wq]/(S[dcI]\NP)$	15	1.3

Table 1: Distribution of *What* categories in questions

mon form of question, and many contain cases of extraction, including some unbounded object extraction. A sample of questions from the corpus is given in Figure 5.

The questions were tokenised according to the Penn Treebank convention and automatically POS tagged. Some of the obvious errors made by the tagger were manually corrected. The first author then manually labelled 500 questions with lexical categories. The supertagger was trained on the annotated questions, and used to label the remaining questions, which were then manually corrected. The performance of the supertagger was good enough at this stage to significantly reduce the effort required for annotation. The second author has verified a subset of the annotated sentences. The question corpus took less than a week to create.

Figure 6 gives the derivations for some example questions. The lexical categories, which make up the annotation in the question corpus, are in bold. Note the first example contains an unbounded object extraction, indicated by the question clause missing an object ( $S[q]/NP$ ) which is an argument of *What*. Table 1 gives the distribution of categories assigned to the first word *What* in each question in the corpus. The first row gives the category of object question *What*. The second row is the object question determiner. The third row is the subject question determiner. And

<sup>4</sup>An earlier version of our QA system used RASP (Briscoe and Carroll, 2002) to parse the questions, but this parser also performed extremely poorly on some question types.

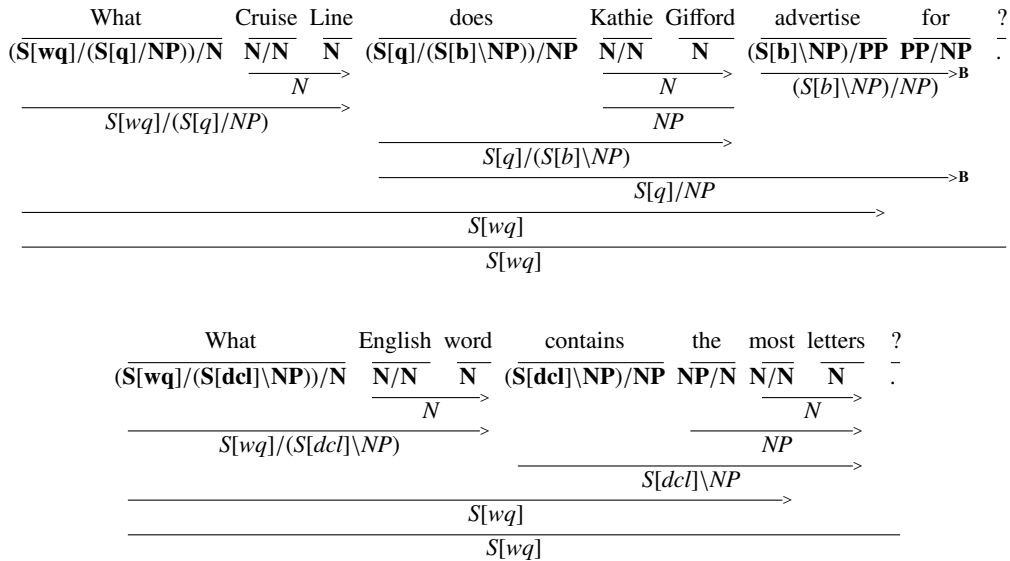


Figure 6: Derivations for example *What*-questions; lexical categories are in bold

the final row is the root subject question *What*. For the examples in Figure 5,  $S[wq]/(S[q]/NP)$  appears in questions 1–6,  $(S[wq]/(S[q]/NP))/N$  in 7–11,  $(S[wq]/(S[dcl]\backslash NP))/N$  in 12–13, and  $S[wq]/(S[dcl]\backslash NP)$  in 14.

## 6 Evaluation

A development set was created by randomly selecting 171 questions. For development purposes the remaining 1,000 questions were used for training; these were also used as a final cross-validation training/test set. The average length of the tokenised questions in the whole corpus is 8.6 tokens.

The lexical category set used by the parser contains all categories which occur at least 10 times in CCGbank, giving a set of 409 categories. In creating the question corpus we used a small number of new category types, of which 3 were needed to cover common question constructions. One of these,  $(S[wq]/(S[dcl]\backslash NP))/N$ , applies to *What*, as in the second example in Figure 6. This category does appear in CCGbank, but so infrequently that it is not part of the parser’s lexical category set. Two more apply to question words like *did* and *is*; for example,  $(S[q]/(S[pss]\backslash NP))/NP$  applies to *is* in *What instrument is Ray Charles best known for playing?*, and  $(S[q]/PP)/NP$  applies to *is* in *What city in Florida is Sea World in?*

### 6.1 Supertagger Accuracy

As an initial evaluation we tested the accuracy of just the supertagger on the development data. The supertagger was run in two modes: one in which a single category was assigned to each word, and one in which 1.5 categories were assigned to each

ACCURACY:	1 CAT		1.5 CATS	
	WORD	SENT	WORD	SENT
MODEL				
CCGbank	72.0	2	84.8	11
Qs	92.3	67	96.6	81
Qs+CCGbank	93.1	61	98.1	87
10Qs+CCGbank	93.6	67	97.9	83

Table 2: Accuracy of supertagger on dev question data

word, on average. Table 2 gives the per-word accuracy on the development question data for a number of supertagging models; SENT accuracy gives the percentage of sentences for which every word is assigned the correct category. Four supertagging models were used: one trained on CCGbank only; one trained on the 1,000 questions; one trained on the 1,000 questions plus CCGbank; and one trained on 10 copies of the 1,000 questions plus CCGbank.

The supertagger performs well when trained on the question data, and benefits from a combination of the questions and CCGbank. To increase the influence of the questions, we tried adding 10 copies of the question data to CCGbank, but this had little impact on accuracy. However, the supertagger performs extremely poorly when trained only on CCGbank. One reason for the very low SENT accuracy figure is that many of the questions contain lexical categories which are not in the supertagger’s category set derived from CCGbank: 56 of the 171 development questions have this property.

The parsing results in Clark and Curran (2004b) rely on a supertagger per-word accuracy of at least 97%, and a sentence accuracy of at least 60% (for 1.5 categories per word). Thus the sentence accu-

SUPERTAGGING / PARSING METHOD	ACCURACY		
	WORD	SENT	WHAT
Increasing av. cats	94.6	82	91
Decreasing av. cats	89.7	65	80
Increasing cats (rand)	93.4	79	88
Decreasing cats (rand)	64.0	9	21
Baseline	68.5	0	61

Table 3: Parser category accuracy on dev data

racy of 11% confirms that our parsing system based only on CCGbank is quite inadequate for accurate question parsing.

## 6.2 Parser Accuracy

Since the gold-standard question data is only labelled at the lexical category level, we are only able to perform a full evaluation at that level. However, the scores in Clark and Curran (2004b) give an indication of how supertagging accuracy corresponds to overall dependency recovery. In addition, in Section 6.3 we present an evaluation on object extraction dependencies in the development data.

We applied the parser to the 171 questions in the development data, using the supertagger model from the third row in Table 2, together with a log-linear parsing model trained on CCGbank. We used the supertagging approach described in Section 2.1, in which a small number of categories is initially assigned to each word, and the parser requests more categories if a spanning analysis cannot be found. We used 4 different values for the parameter  $\beta$  (which determines the average number of categories per word): 0.5, 0.25, 0.075 and 0.01. The average number of categories at each level for the development data is 1.1, 1.2, 1.6 and 3.8. The parser provided an analysis for all but one of the 171 questions.

The first row of Table 3 gives the per-word, and sentence, category accuracy for the parser output. Figures are also given for the accuracy of the categories assigned to the first word *What*. The figures show that the parser is more accurate at supertagging than the single-category supertagger.

The second row gives the results if the original supertagging approach of Clark et al. (2002) is used, i.e. starting with a high number of categories per word, and reducing the number if the sentence cannot be parsed within reasonable space and time constraints. The third row corresponds to our new supertagging approach, but chooses a derivation at random, by randomly traversing the packed chart representation used by the parser. The fourth row corresponds to the supertagging approach of Clark et al. (2002), together with a random selection of

SUPERTAGGING / PARSING METHOD	ACCURACY		
	WORD	SENT	WHAT
Increasing av. cats	94.4	79	92
Decreasing av. cats	89.5	64	81

Table 4: Cross-validation results

the derivation. The baseline method in the fifth row assigns to a word the category most frequently seen with it in the data; for unseen words  $N$  is assigned.

The results in Table 3 demonstrate that our new supertagging approach is very effective. The reason is that the parser typically uses the first supertagger level, where the average number of categories per word is only 1.1, and the per-word/sentence category accuracies are 95.5 and 70.8%, respectively. 136 of the 171 questions (79.5%) are parsed at this level. Since the number of categories per word is very small, the parser has little work to do in combining the categories; the supertagger is effectively an *almost-parser* (Bangalore and Joshi, 1999). Thus the parsing model, which is not tuned for questions, is hardly used by the parser. This interpretation is supported by the high scores for the random method in row 3 of the table.

In contrast, the previous supertagging method of Clark et al. (2002) results in a large derivation space, which must be searched using the parsing model. Thus the accuracy of the parser is greatly reduced, as shown in rows 2 and 4.

As a final test of the robustness of our results, we performed a cross-validation experiment using the 1,000 training questions. The 1,000 questions were randomly split into 10 chunks. Each chunk was used as a test set in a separate run, with the remaining chunks as training data plus CCGbank. Table 4 gives the results averaged over the 10 runs for the two supertagging approaches.

## 6.3 Object Extraction in Questions

For the object extraction evaluation we considered the 36 questions in the development data which have the category  $(S[wq]/(S[q]/NP))/N$  assigned to *What*. Table 7 gives examples of the questions. We assume these are fairly representative of the kinds of object extraction found in other question types, and thus present a useful test set.

We parsed the questions using the best performing configuration from the previous section. All but one of the sentences was given an analysis. The per-word/sentence category accuracies were 90.2% and 71.4%, respectively. These figures are lower than for the corpus as a whole, suggesting these object extraction questions are more difficult than average.

What amount of folic acid should an expectant mother take daily?  
What movie did Madilyn Kahn star in with Gene Wilder?  
What continent is Egypt on?  
What year was Ebbets Field, home of Brooklyn Dodgers, built?  
What body of water does the Colorado River empty into?

Figure 7: Examples of object extraction questions

We inspected the output to see if the object dependencies had been recovered correctly. To get the object dependency correct in the first question in Table 7, for example, the parser would need to assign the correct category to *take* and return *amount* as the object of *take*. Of the 37 extracted object dependencies (one question had two such dependencies), 29 (78.4%) were recovered correctly. Given that the original parser trained on CCGbank performs extremely poorly on such questions, we consider this to be a highly promising result.

## 7 Conclusion

We have presented a detailed evaluation of a CCG parser on object extraction dependencies in WSJ text. Given the difficulty of the task, the accuracy of the parser is encouraging. The errors made by the parser suggest that general improvements in the coverage of the lexicon and parsing models derived from CCGbank will lead to improved recovery of long-range object dependencies.

In contrast, we have suggested that general improvements in CCGbank parsing models will not lead to satisfactory performance on question parsing. The reason is that the *Wh*-question domain is syntactically distinct from WSJ text. We have presented a novel method for porting the parser to the question domain, which has led to good performance on question parsing. This has also demonstrated the close integration of the supertagger and the CCG parser on which our method depends.

One of the major drawbacks of current NLP technology is that in general it performs very poorly outside of the training data domain. Our porting method only requires lexical category data, which is far easier to produce than full parse trees. This is an efficient method for porting the parser to other domains. The method may also be applicable to other lexicalised grammar formalisms.

We will extend the question corpus to other question types. We are also continuing to develop the supertagger, which we have demonstrated is central to efficient portable wide-coverage CCG parsing.

## References

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

- Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING-04*, Geneva, Switzerland.
- Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd LREC Conference*, pages 1499–1504, Las Palmas, Gran Canaria.
- Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the NAACL*, pages 132–139, Seattle, WA.
- Stephen Clark and James R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of COLING-04*, Geneva, Switzerland.
- Stephen Clark and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Meeting of the ACL*, Barcelona, Spain.
- Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Meeting of the ACL*, pages 327–334, Philadelphia, PA.
- Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.
- Peter Dienes and Amit Dubey. 2003. Deep syntactic processing by combining shallow methods. In *Proceedings of the EMNLP Conference*, pages 431–438, Sapporo, Japan.
- C. Doran, B. Hockey, P. Hopely, J. Rosenzweig, A. Sarkar, B. Srinivas, F. Xia, A. Nasr, and O. Rambow. 1997. Maintaining the forest and burning out the underbrush in XTAG. In *Proceedings of the EN-VGRAM Workshop*, Madrid, Spain.
- Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorical Grammar. In *Proceedings of the 40th Meeting of the ACL*, pages 335–342, Philadelphia, PA.
- Julia Hockenmaier. 2003a. *Data and Models for Statistical Parsing with Combinatory Categorical Grammar*. Ph.D. thesis, University of Edinburgh.
- Julia Hockenmaier. 2003b. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Meeting of the ACL*, pages 359–366, Sapporo, Japan.
- Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Meeting of the ACL*, pages 136–143, Philadelphia, PA.
- Marius Pasca and Sanda Harabagiu. 2001. High performance question/answering. In *Proceedings of the ACL SIGIR Conference on Research and Development in Information Retrieval*, pages 366–374, New Orleans LA.
- Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press, Cambridge, MA.
- Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.