QuestionBank: Creating a Corpus of Parse-Annotated Questions

John Judge[1], **Aoife Cahill**[1], **and Josef van Genabith**[1,2]
[1]National Centre for Language Technology and School of Computing,
Dublin City University, Dublin, Ireland
[2]IBM Dublin Center for Advanced Studies,
IBM Dublin, Ireland
{jjudge,acahill,josef}@computing.dcu.ie

Abstract

This paper describes the development of QuestionBank, a corpus of 4000 parse-annotated questions for (i) use in training parsers employed in QA, and (ii) evaluation of question parsing. We present a series of experiments to investigate the effectiveness of QuestionBank as both an exclusive and supplementary training resource for a state-of-the-art parser in parsing both question and non-question test sets. We introduce a new method for recovering empty nodes and their antecedents (capturing long distance dependencies) from parser output in CFG trees using LFG f-structure reentrancies. Our main findings are (i) using QuestionBank training data improves parser performance to 89.75% labelled bracketing f-score, an increase of almost 11% over the baseline; (ii) back-testing experiments on non-question data (Penn-II WSJ Section 23) shows that the retrained parser does not suffer a performance drop on non-question material; (iii) ablation experiments show that the size of training material provided by QuestionBank is sufficient to achieve optimal results; (iv) our method for recovering empty nodes captures long distance dependencies in questions from the ATIS corpus with high precision (96.82%) and low recall (39.38%). In summary, QuestionBank provides a useful new resource in parser-based QA research.

1 Introduction

Parse-annotated corpora (treebanks) are crucial for developing machine learning and statistics-based parsing resources for a given language or task. Large treebanks are available for major languages, however these are often based on a specific text type or genre, e.g. financial newspaper text (the Penn-II Treebank (Marcus et al., 1993)). This can limit the applicability of grammatical resources induced from treebanks in that such resources underperform when used on a different type of text or for a specific task.

In this paper we present work on creating QuestionBank, a treebank of parse-annotated questions, which can be used as a supplementary training resource to allow parsers to accurately parse questions (as well as other text). Alternatively, the resource can be used as a stand-alone training corpus to train a parser specifically for questions. Either scenario will be useful in training parsers for use in question answering (QA) tasks, and it also provides a suitable resource to evaluate the accuracy of these parsers on questions.

We use a semi-automatic "bootstrapping" method to create the question treebank from raw text. We show that a parser trained on the question treebank alone can accurately parse questions. Training on a combined corpus consisting of the question treebank and an established training set (Sections 02-21 of the Penn-II Treebank), the parser gives state-of-the-art performance on both questions and a non-question test set (Section 23 of the Penn-II Treebank).

Section 2 describes background work and motivation for the research presented in this paper. Section 3 describes the data we used to create the corpus. In Section 4 we describe the semi-automatic method to "bootstrap" the question corpus, discuss some interesting and problematic phenomena, and show how the manual vs. automatic workload distribution changed as work progressed. Two sets of experiments using our new question corpus are presented in Section 5. In Section 6 we introduce a new method for recovering empty nodes and their antecedents using Lexical Functional Grammar (LFG) f-structure reen-

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 497–504,
Sydney, July 2006. ©2006 Association for Computational Linguistics

trancies. Section 7 concludes and outlines future work.

2 Background and Motivation

High quality probabilistic, treebank-based parsing resources can be rapidly induced from appropriate treebank material. However, treebank- and machine learning-based grammatical resources reflect the characteristics of the training data. They generally underperform on test data substantially different from the training data.

Previous work on parser performance and domain variation by Gildea (2001) showed that by training a parser on the Penn-II Treebank and testing on the Brown corpus, parser accuracy drops by 5.7% compared to parsing the Wall Street Journal (WSJ) based Penn-II Treebank Section 23. This shows a negative effect on parser performance even when the test data is *not radically* different from the training data (both the Penn II and Brown corpora consist primarily of written texts of American English, the main difference is the considerably more varied nature of the text in the Brown corpus). Gildea also shows how to resolve this problem by adding appropriate data to the training corpus, but notes that a large amount of additional data has little impact if it is not matched to the test material.

Work on more *radical* domain variance and on adapting treebank-induced LFG resources to analyse ATIS (Hemphill et al., 1990) question material is described in Judge et al. (2005). The research established that even a small amount of additional training data can give a substantial improvement in question analysis in terms of both CFG parse accuracy and LFG grammatical functional analysis, with no significant negative effects on non-question analysis. Judge et al. (2005) suggest, however, that further improvements are possible given a larger question training corpus.

Clark et al. (2004) worked specifically with question parsing to generate dependencies for QA with Penn-II treebank-based Combinatory Categorial Grammars (CCG's). They use "what" questions taken from the TREC QA datasets as the basis for a What-Question corpus with CCG annotation.

3 Data Sources

The raw question data for QuestionBank comes from two sources, the TREC 8-11 QA track test sets[1], and a question classifier training set produced by the Cognitive Computation Group (CCG[2]) at the University of Illinois at Urbana-Champaign.[3] We use equal amounts of data from each source so as not to bias the corpus to either data source.

3.1 TREC Questions

The TREC evaluations have become the standard evaluation for QA systems. Their test sets consist primarily of fact seeking questions with some imperative statements which request information, e.g. "List the names of cell phone manufacturers." We included 2000 TREC questions in the raw data from which we created the question treebank. These 2000 questions consist of the test questions for the first three years of the TREC QA track (1893 questions) and 107 questions from the 2003 TREC test set.

3.2 CCG Group Questions

The CCG provide a number of resources for developing QA systems. One of these resources is a set of 5500 questions and their answer types for use in training question classifiers. The 5500 questions were stripped of answer type annotation, duplicated TREC questions were removed and 2000 questions were used for the question treebank.

The CCG 5500 questions come from a number of sources (Li and Roth, 2002) and some of these questions contain minor grammatical mistakes so that, in essence, this corpus is more representative of genuine questions that would be put to a working QA system. A number of changes in tokenisation were corrected (eg. separating contractions), but the minor grammatical errors were left unchanged because we believe that it is necessary for a parser for question analysis to be able to cope with this sort of data if it is to be used in a working QA system.

4 Creating the Treebank

4.1 Bootstrapping a Question Treebank

The algorithm used to generate the question treebank is an iterative process of parsing, manual correction, retraining, and parsing.

[1] http://trec.nist.gov/data/qa.html

[2] Note that the acronym CCG here refers to Cognitive Computation Group, rather than Combinatory Categorial Grammar mentioned in Section 2.

[3] http://l2r.cs.uiuc.edu/ cogcomp/tools.php

Algorithm 1 Induce a parse-annotated treebank from raw data

repeat
  Parse a new section of raw data
  Manually correct errors in the parser output
  Add the corrected data to the training set
  Extract a new grammar for the parser
until All the data has been processed

Algorithm 1 summarises the bootstrapping algorithm. A section of raw data is parsed. The parser output is then manually corrected, and added to the parser's training corpus. A new grammar is then extracted, and the next section of raw data is parsed. This process continues until all the data has been parsed and hand corrected.

4.2 Parser

The parser used to process the raw questions prior to manual correction was that of Bikel (2002)[4], a retrainable emulation of Collins (1999) model 2 parser. Bikel's parser is a history-based parser which uses a lexicalised generative model to parse sentences. We used WSJ Sections 02-21 of the Penn-II Treebank to train the parser for the first iteration of the algorithm. The training corpus for subsequent iterations consisted of the WSJ material and increasing amounts of processed questions.

4.3 Basic Corpus Development Statistics

Our question treebank was created over a period of three months at an average annotation speed of about 60 questions per day. This is quite rapid for treebank development. The speed of the process was helped by two main factors: the questions are generally quite short (typically about 10 words long), and, due to retraining on the continually increasing training set, the quality of the parses output by the parser improved dramatically during the development of the treebank, with the effect that corrections during the later stages were generally quite small and not as time consuming as during the initial phases of the bootstrapping process.

For example, in the first week of the project the trees from the parser were of relatively poor quality and over 78% of the trees needed to be corrected manually. This slowed the annotation process considerably and parse-annotated questions were being produced at an average rate of 40 trees per day. During the later stages of the project this had changed dramatically. The quality of trees from the parser was much improved with less than 20% of the trees requiring manual correction. At this stage parse-annotated questions were being produced at an average rate of 90 trees per day.

[4]Downloaded from http://www.cis.upenn.edu/∼dbikel/software.html#stat-parser

4.4 Corpus Development Error Analysis

Some of the more frequent errors in the parser output pertain to the syntactic analysis of WH-phrases (WHNP, WHPP, etc). In Sections 02-21 of the Penn-II Treebank, these are used more often in relative clause constructions than in questions. As a result many of the corpus questions were given syntactic analyses corresponding to relative clauses (SBAR with an embedded S) instead of as questions (SBARQ with an embedded SQ). Figure 1 provides an example.

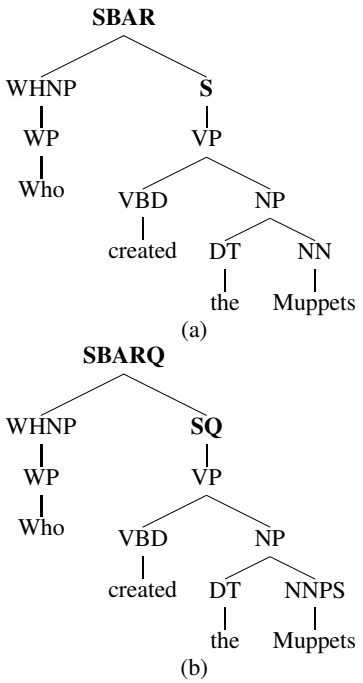

Figure 1: Example tree before (a) and after correction (b)

Because the questions are typically short, an error like this has quite a large effect on the accuracy for the overall tree; in this case the f-score for the parser output (Figure 1(a)) would be only 60%. Errors of this nature were quite frequent in the first section of questions analysed by the parser, but with increased training material becoming available during successive iterations, this error became less frequent and towards the end of

the project it was only seen in rare cases.

WH-XP marking was the source of a number of consistent (though infrequent) errors during annotation. This occurred mostly in PP constructions containing WHNPs. The parser would output a structure like Figure 2(a), where the PP mother of the WHNP is not correctly labelled as a WHPP as in Figure 2(b).

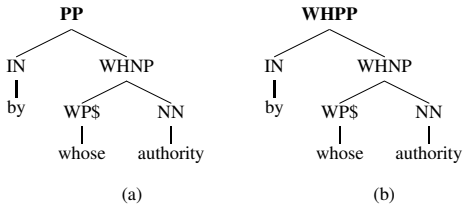

Figure 2: WH-XP assignment

The parser output often had to be rearranged structurally to varying degrees. This was common in the longer questions. A recurring error in the parser output was failing to identify VPs in SQs with a single object NP. In these cases the verb and the object NP were left as daughters of the SQ node. Figure 3(a) illustrates this, and Figure 3(b) shows the corrected tree with the VP node inserted.

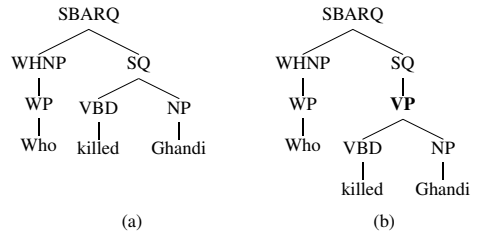

Figure 3: VP missing inside SQ with a single NP

On inspection, we found that the problem was caused by copular constructions, which, according to the Penn-II annotation guidelines, do not feature VP constituents. Since almost half of the question data contain copular constructions, the parser trained on this data would sometimes misanalyse non-copular constructions or, conversely, incorrectly bracket copular constructions using a VP constituent (Figure 4(a)).

The predictable nature of these errors meant that they were simple to correct. This is due to the particular context in which they occur and the finite number of forms of the copular verb.

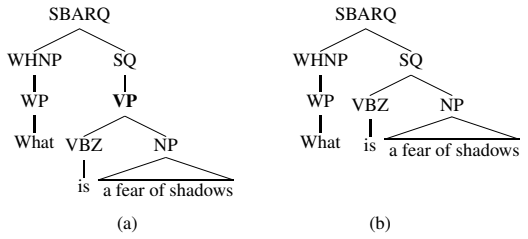

Figure 4: Erroneous VP in copular constructions

5 Experiments with QuestionBank

In order to test the effect training on the question corpus has on parser performance, we carried out a number of experiments. In cross-validation experiments with 90%/10% splits we use all 4000 trees in the completed QuestionBank as the test set. We performed ablation experiments to investigate the effect of varying the amount of question and non-question training data on the parser's performance. For these experiments we divided the 4000 questions into two sets. We randomly selected 400 trees to be held out as a gold standard test set against which to evaluate, the remaining 3600 trees were then used as a training corpus.

5.1 Establishing the Baseline

The baseline we use for our experiments is provided by Bikel's parser trained on WSJ Sections 02-21 of the Penn-II Treebank. We test on all 4000 questions in our question treebank, and also Section 23 of the Penn-II Treebank.

QuestionBank		WSJ Section 23	
Coverage	100	Coverage	100
F-Score	78.77	F-Score	82.97

Table 1: Baseline parsing results

Table 1 shows the results for our baseline evaluations on question and non-question test sets. While the coverage for both tests is high, the parser underperforms significantly on the question test set with a labelled bracketing f-score of 78.77 compared to 82.97 on Section 23 of the Penn-II Treebank. Note that unlike the published results for Bikel's parser in our evaluations we test on Section 23 and *include punctuation*.

5.2 Cross-Validation Experiments

We carried out two cross-validation experiments. In the first experiment we perform a 10-fold cross-validation experiment using our 4000 question

treebank. In each case a randomly selected set of 10% of the questions in QuestionBank was held out during training and used as a test set. In this way parses from unseen data were generated for all 4000 questions and evaluated against the QuestionBank trees.

The second cross-validation experiment was similar to the first, but in each of the 10 folds we train on 90% of the 4000 questions in QuestionBank and on all of Sections 02-21 of the Penn-II Treebank.

In both experiments we also backtest each of the ten grammars on Section 23 of the Penn-II Treebank and report the average scores.

QuestionBank		Backtest on Sect 23	
Coverage	100	Coverage	98.79
F-Score	88.82	F-Score	59.79

Table 2: Cross-validation experiment using the 4000 question treebank

Table 2 shows the results for the first cross-validation experiment, using only the 4000 sentence QuestionBank. Compared to Table 1, the results show a significant improvement of over 10% on the baseline f-score for questions. However, the tests on the non-question Section 23 data show not only a significant drop in accuracy but also a drop in coverage.

Questions		Backtest on Sect 23	
Coverage	100	Coverage	100
F-Score	89.75	F-Score	82.39

Table 3: Cross-validation experiment using Penn-II Treebank Sections 02-21 and 4000 questions

Table 3 shows the results for the second cross-validation experiment using Sections 02-21 of the Penn-II Treebank and the 4000 questions in QuestionBank. The results show an even greater increase on the baseline f-score than the experiments using only the question training set (Table 2). The non-question results are also better and are comparable to the baseline (Table 1).

5.3 Ablation Runs

In a further set of experiments we investigated the effect of varying the amount of data in the parser's training corpus. We experiment with varying both the amount of QuestionBank and Penn-II Treebank data that the parser is trained on. In each experiment we use the 400 question test set and Section 23 of the Penn-II Treebank to evaluate against, and the 3600 question training set described above and Sections 02-21 of the Penn-II Treebank as the basis for the parser's training corpus. We report on three experiments:

In the first experiment we train the parser using only the 3600 question training set. We performed ten training and parsing runs in this experiment, incrementally reducing the size of the QuestionBank training corpus by 10% of the whole on each run.

The second experiment is similar to the first but in each run we add Sections 02-21 of the Penn-II Treebank to the (shrinking) training set of questions.

The third experiment is the converse of the second, the amount of questions in the training set remains fixed (all 3600) and the amount of Penn-II Treebank material is incrementally reduced by 10% on each run.

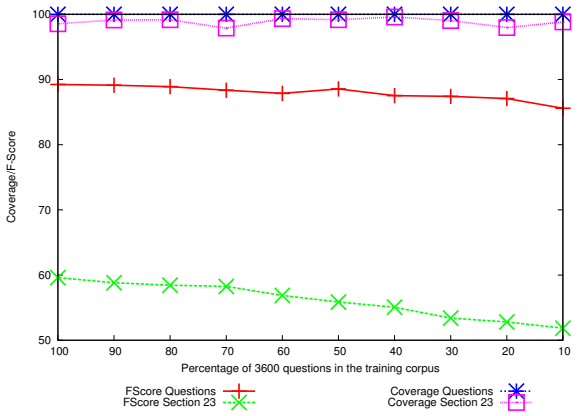

Figure 5: Results for ablation experiment reducing 3600 training questions in steps of 10%

Figure 5 graphs the coverage and f-score for the parser in tests on the 400 question test set, and Section 23 of the Penn-II Treebank in ten parsing runs with the amount of data in the 3600 question training corpus reducing incrementally on each run. The results show that training on only a small amount of questions, the parser can parse questions with high accuracy. For example when trained on only 10% of the 3600 questions used in this experiment, the parser successfully parses all of the 400 question test set and achieves an f-score of 85.59. However the results for the tests on WSJ Section 23 are considerably worse. The parser never manages to parse the full test set, and the best score at 59.61 is very low.

Figure 6 graphs the results for the second abla-

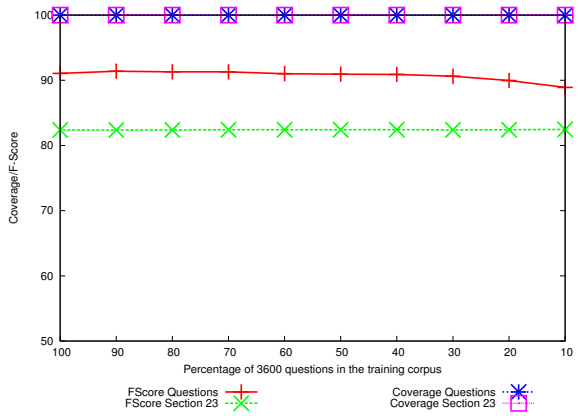

Figure 6: Results for ablation experiment using PTB Sections 02-21 (fixed) and reducing 3600 questions in steps of 10%

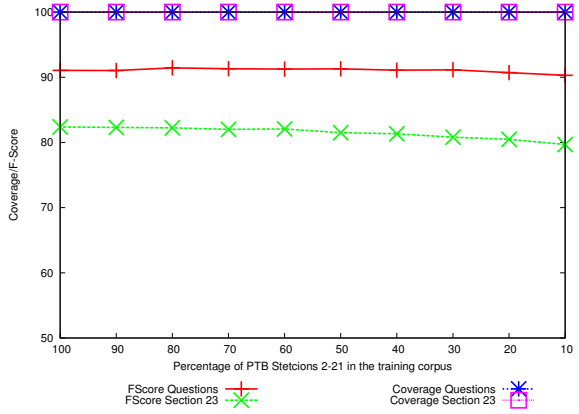

Figure 7: Results for ablation experiment using 3600 questions (fixed) and reducing PTB Sections 02-21 in steps of 10%

tion experiment. The training set for the parser consists of a fixed amount of Penn-II Treebank data (Sections 02-21) and a reducing amount of question data from the 3600 question training set. Each grammar is tested on both the 400 question test set, and WSJ Section 23. The results here are significantly better than in the previous experiment. In all of the runs the coverage for both test sets is 100%, f-scores for the question test set decrease as the amount of question data in the training set is reduced (though they are still quite high.) There is little change in the f-scores for the tests on Section 23, the results all fall in the range 82.36 to 82.46, which is comparable to the baseline score.

Figure 7 graphs the results for the third ablation experiment. In this case the training set is a fixed amount of the question training set described above (all 3600 questions) and a reducing amount of data from Sections 02-21 of the Penn Treebank. The graph shows that the parser performs consistently well on the question test set in terms of both coverage and accuracy. The tests on Section 23, however, show that as the amount of Penn-II Treebank material in the training set decreases, the f-score also decreases.

6 Long Distance Dependencies

Long distance dependencies are crucial in the proper analysis of question material. In English wh-questions, the fronted wh-constituent refers to an argument position of a verb inside the interrogative construction. Compare the superficially similar

1. Who$_1$ $[t_1]$ killed Harvey Oswald?

2. Who$_1$ did Harvey Oswald kill $[t_1]$?

(1) queries the agent (syntactic subject) of the described eventuality, while (2) queries the patient (syntactic object). In the Penn-II and ATIS treebanks, dependencies such as these are represented in terms of empty productions, traces and coindexation in CFG tree representations (Figure 8).

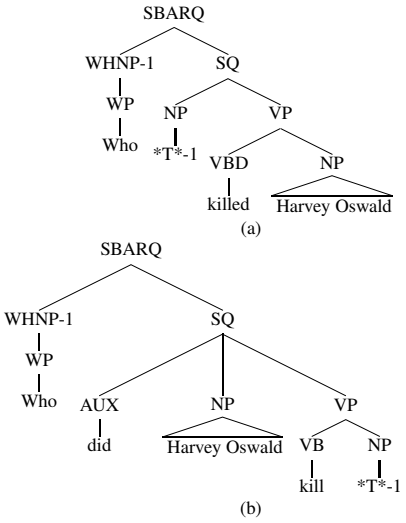

Figure 8: LDD resolved treebank style trees

With few exceptions[5] the trees produced by current treebank-based probabilistic parsers do not represent long distance dependencies (Figure 9).

Johnson (2002) presents a tree-based method for reconstructing LDD dependencies in Penn-II trained parser output trees. Cahill et al. (2004) present a method for resolving LDDs

[5]Collins' Model 3 computes a limited number of wh-dependencies in relative clause constructions.

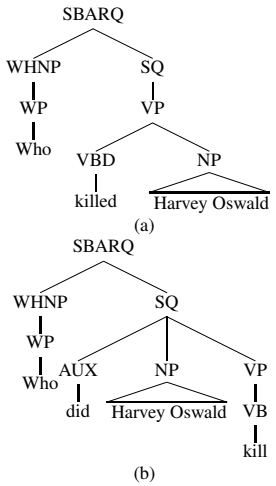

Figure 9: Parser output trees

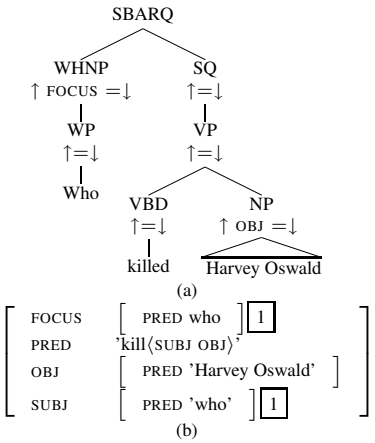

Figure 10: Annotated tree and f-structure

at the level of Lexical-Functional Grammar f-structure (attribute-value structure encodings of basic predicate-argument structure or dependency relations) without the need for empty productions and coindexation in parse trees. Their method is based on learning finite approximations of functional uncertainty equations (regular expressions over paths in f-structure) from an automatically f-structure annotated version of the Penn-II treebank and resolves LDDs at f-structure. In our work we use the f-structure-based method of Cahill et al. (2004) to "reverse engineer" empty productions, traces and coindexation in parser output trees. We explain the process by way of a worked example.

We use the parser output tree in Figure 9(a) (without empty productions and coindexation) and automatically annotate the tree with f-structure information and compute LDD-resolution at the level of f-structure using the resources of Cahill et al. (2004). This generates the f-structure annotated tree[6] and the LDD resolved f-structure in Figure 10.

Note that the LDD is indicated in terms of a reentrancy $\boxed{1}$ between the question FOCUS and the SUBJ function in the resolved f-structure. Given the correspondence between the f-structure and f-structure annotated nodes in the parse tree, we compute that the SUBJ function newly introduced and reentrant with the FOCUS function is an argument of the PRED 'kill' and the verb form 'killed' in the tree. In order to reconstruct the corresponding empty subject NP node in the parser output tree, we need to determine candidate anchor sites for the empty node. These anchor sites can only be realised along the path up to the maximal projection of the governing verb indicated by $\uparrow{=}\downarrow$ annotations in LFG. This establishes three anchor sites: VP, SQ and the top level SBARQ. From the automatically f-structure annotated Penn-II treebank, we extract f-structure annotated PCFG rules for each of the three anchor sites whose RHSs contain exactly the information (daughter categories plus LFG annotations) in the tree in Figure 10 (in the same order) plus an additional node (of whatever CFG category) annotated $\uparrow$SUBJ$=\downarrow$, located anywhere within the RHSs. This will retrieve rules of the form

$\text{VP} \rightarrow NP[\uparrow \text{SUBJ} = \downarrow]\ VBD[\uparrow{=}\downarrow]\ NP[\uparrow \text{OBJ} = \downarrow]$
$VP \rightarrow \ldots$
$\ldots$
$SQ \rightarrow NP[\uparrow \text{SUBJ} = \downarrow]\ VP[\uparrow{=}\downarrow]$
$SQ \rightarrow \ldots$
$\ldots$
$SBARQ \rightarrow \ldots$
$\ldots$

each with their associated probabilities. We select the rule with the highest probability and cut the rule into the tree in Figure 10 at the appropriate anchor site (as determined by the rule LHS). In our case this selects $SQ \rightarrow NP[\uparrow \text{SUBJ}{=}\downarrow]VP[\uparrow{=}\downarrow]$ and the resulting tree is given in Figure 11. From this tree, it is now easy to compute the tree with the coindexed trace in Figure 8 (a).

In order to evaluate our empty node and coindexation recovery method, we conducted two experiments, one using 146 gold-standard ATIS question trees and one using parser output on the corresponding strings for the 146 ATIS question trees.

[6] Lexical annotations are suppressed to aid readability.

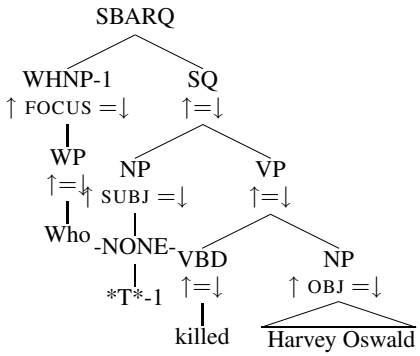

Figure 11: Resolved tree

In the first experiment, we delete empty nodes and coindexation from the ATIS gold standard trees and and reconstruct them using our method and the preprocessed ATIS trees. In the second experiment, we parse the strings corresponding to the ATIS trees with Bikel's parser and reconstruct the empty productions and coindexation. In both cases we evaluate against the original (unreduced) ATIS trees and score if and only if all of insertion site, inserted CFG category and coindexation match.

	Parser Output	Gold Standard Trees
Precision	96.77	96.82
Recall	38.75	39.38

Table 4: Scores for LDD recovery (empty nodes and antecedents)

Table 4 shows that currently the recall of our method is quite low at 39.38% while the accuracy is very high with precision at 96.82% on the ATIS trees. Encouragingly, evaluating parser output for the same sentences shows little change in the scores with recall at 38.75% and precision at 96.77%.

7 Conclusions

The data represented in Figure 5 show that training a parser on 50% of QuestionBank achieves an f-score of 88.56% as against 89.24% for training on all of QuestionBank. This implies that while we have not reached an absolute upper bound, the question corpus is sufficiently large that the gain in accuracy from adding more data is so small that it does not justify the effort.

We will evaluate grammars learned from QuestionBank as part of a working QA system. A beta-release of the non-LDD-resolved QuestionBank is available for download at `http://www.computing.dcu.ie/~jjudge/qtreebank/4000qs.txt`. The final, hand-corrected, LDD-resolved version will be available in October 2006.

Acknowledgments

We are grateful to the anonymous reviewers for their comments and suggestions. This research was supported by Science Foundation Ireland (SFI) grant 04/BR/CS0370 and an Irish Research Council for Science Engineering and Technology (IRCSET) PhD scholarship 2002-05.

References

Daniel M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of HLT 2002*, pages 24–27, San Diego, CA.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations. In *Proceedings of ACL-04*, pages 320–327, Barcelona, Spain.

Stephen Clark, Mark Steedman, and James R. Curran. 2004. Object-extraction and question-parsing using ccg. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP-04*, pages 111–118, Barcelona, Spain.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Daniel Gildea. 2001. Corpus variation and parser performance. In Lillian Lee and Donna Harman, editors, *Proceedings of EMNLP*, pages 167–202, Pittsburgh, PA.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS Spoken Language Systems pilot corpus. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 96–101, Hidden Valley, PA.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings ACL-02*, University of Pennsylvania, Philadelphia, PA.

John Judge, Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2005. Strong Domain Variation and Treebank-Induced LFG Resources. In *Proceedings LFG-05*, pages 186–204, Bergen, Norway, July.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of COLING-02*, pages 556–562, Taipei, Taiwan.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, **19**(2):313–330.