Adapting and Developing Linguistic Resources for Question Answering

John Judge

A dissertation submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy

to the

DCU

Dublin City University

School of Computing

Supervisors: Prof. Josef van Genabith
Dr. Aoife Cahill

December 2006

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work

Signed ______________________

(John Judge)

Student ID 98461796

Date 22nd December 2006

Contents

Abstract

As information retrieval becomes more focussed, so too must the techniques involved in the retrieval process. More precise responses to queries require more precise linguistic analysis of both the queries and the factual documents from which the information is being retrieved.

In this thesis, I present research into using existing linguistic tools to analyse questions. These tools, as supplied, often underperform on question analysis. I present my work on adapting these tools, and creating new resources for use in developing new tools tailored to question analysis.

My work has shown that in order to adapt the treebank- and f-structure annotation algorithm-based wide coverage LFG parsing resources of Cahill et al. (2004) to analyse questions from the ATIS corpus, only the c-structure parser needs to be retrained, the annotation algorithm remains unchanged. The retrained c-structure parser needs only a small amount of appropriate training data added to its training corpus to gain a significant improvement in both c-structure parsing and f-structure annotation.

Given the improvements made with a relatively small amount of question data, I developed QuestionBank, a question treebank, to determine what further gains can be made using a larger amount of question data. My question treebank is a corpus of 4000 parse annotated questions. The questions were taken from a number of sources and the question treebank was "bootstrapped" in an incremental parsing, hand correction and retraining approach from raw data using existing probabilistic parsing resources.

Experiments with QuestionBank show that it is an effective resource for training parsers to analyse questions with an improvement of over 10% on the baseline parsing results. In further experiments I show that a parser retrained with QuestionBank can also parse newspaper text (Penn-II Treebank Section 23) with state-of-the-art accuracy.

Long distance dependencies (LDDs) are a vital part of question analysis in determining semantic roles and question focus. I have designed and implemented a novel method to recover WH-traces and coindexed antecedents in c-structure trees from parser output which uses the f-structure LDD resolution method of Cahill et al (2004) to resolve the dependencies and then "reverse engineers" the corresponding syntactic components in the c-structure tree.

Acknowledgements

I wish to acknowledge everyone who has helped me with my research and this thesis. I would like to thank the Irish Research Council for Science Engineering and Technology and the Embark Initiative for supporting me through their Ph.D. Research Grant scheme. I also thank my examiners for their useful comments, suggestions and feedback on this thesis. I would like to express my deepest gratitude to my supervisors, Josef van Genabith and Aoife Cahill, without whom I would never have come this far.

Josef has always encouraged of me and my work, and even when I had given up hope on an idea he was always there with new insight, advice and a new perspective. I'd especially like to thank Josef for his understanding when I walked into his office asking to take an extended leave of absence to sail the tall ship Jeanie Johnston (http://www.jeaniejohnston.ie) on her maiden voyage to America. The positive reaction towards and enthusiasm for the idea, while typically Josef, was not what I expected and I am eternally grateful to him for helping me take such a once in a lifetime opportunity and still have my research to come back to. The advice, guidance, and all important reality checking Aoife gave me was invaluable. I have learned so much from Aoife and her door was always open to me, even "electronically" when she was away on a 6 month research visit at PARC in California. Her ability to explain a complex topic in simple terms saved me on many an occasion and inspired me to think differently about my own work.

I would also like to thank all of the members of the National Centre for Language Technology and the School of Computing postgrads for providing such a relaxed an enjoyable atmosphere to work in and for listening to presentations about my work and always having helpful feedback. In this regard particular mention must go to Cara Greene and Katrina Keogh for always being there for me.

My time sailing on board the Jeanie Johnston was unforgettable, not only for the experience itself, but also because of the friends I made. These friends have always shown an interest in my academic pursuits and encouraged me in every regard. I'd like to thank all of the crew of the Jeanie especially Lou, Boyce, Keelin, Tom H and Adrian, and of course Captains Mc Carthy and Coleman for making it all possible.

I would also like to thank all of my friends, their names are too numerous to mention here but they have been my support and my distraction from work when I needed it most. I'd like to especially thank Dan and Stuart for proving that out of sight doesn't mean out of mind when it comes to friendship. They have both been there any time day or night no matter where they were to talk to or to drag me from work for my sanity's sake. Stuart has promised never to read my thesis so that he won't have to tell lies about what he thought of its content. Dan I know will read some if not all of this. I hope it gives you an idea of what I have been talking about all this time.

I can never thank Úna enough for her words of encouragement, unquestioning belief in me and for being my rock through the toughest times. I know she knows this, however I feel her contribution to this thesis deserves acknowledgement here. Her skills as a proof reader were also invaluable; I don't think she ever would have imagined becoming such an expert in computational linguistics.

My parents, Mary and Marin, and my extended family have been the greatest source of support and inspiration I could have asked for. While I'm sure they never fully understood what I was doing, or even why, they were always encouraging and excited about any new developments and would listen when I needed to vent steam. All this despite only getting to see me once in a blue moon. Going home was a luxury I rarely afforded myself for fear I'd get used too to it, instead it was my retreat, and my safe haven. My Grandparents have, I think, been the most vocal in their support for my through my (many) years at DCU and while my Nana Pauline is breathing a sigh of relief that I'm finally finished college, I know my Nana Kay is thrilled that her grandson who wanted to be a binman is a doctor just like she said he should be.

"Anything in this life that is worth having is hard to get," is a statement I've heard many times throughout my life when confronted with a difficult task. While I'm not sure which of these said it first (or if it was borrowed from someone else) I must thank my father, and both grandfathers for instilling this idea in me. It has motivated me to keep going. Greater role models than these three men I could never ask for.

List of Tables

List of Figures

List of Acronyms

AI	Artificial Intelligence
APCFG	Annotated Probabilistic Context-Free Grammar
ATIS	Air Travel Information System
AVM	Attribute-Value Matrix
CALL	Computer Aided Language Learning
CCG	Combinatory Categorial Grammar
CCG	Cognitive Computation Group
CF-PSG	Context-Free Phrase Structure Grammar
CFG	Context-Free Grammar
CLIR	Cross Language Information Retrieval
CLQA	Cross Language Question Answering
GF	Grammatical Function
IR	Information Retrieval
LDD	Long-Distance Dependency
LFG	Lexical-Functional Grammar
LHS	Left Hand Side
MMIR	Multimedia Information Retrieval
NCLT	National Centre for Language Technology
NLP	Natural Language Processing
PCFG	Probabilistic Context-Free Grammar
POS	Part Of Speech
QA	Question Answering
RHS	Right Hand Side
SVO	Subject-Verb-Object
TREC	Text Retrieval Conference

VSO	Verb-Subject-Object
WSJ	Wall Street Journal
WWW	Worldwide Web

Chapter 1

Introduction

Question Answering (QA), the process of retrieving precise information that satisfies a user query in the form of a question, is an inherently more linguistically involved task than document-based Information Retrieval (IR). Linguistic analysis of the structure and functional roles of a question can identify important information such as the question focus and how it relates to the main predicate of the question. Linguistic analysis is useful for disambiguating strings which are similar on a surface level but have quite different meaning. Such distinctions can be lost by systems that only look at the surface string and perform little or no deeper linguistic analysis.

To date, linguistic analysis has been employed to different degrees in QA. The current trend in state-of-the-art QA systems is to use deeper semantic, logical form or predicate-argument structures derived from the natural language strings for both documents and queries. These structures are then used for a range of tasks from answer verification to inference. The ways and means by which these representations of linguistic data are employed and derived depends on the system, but one thing that is common among the NLP-rich systems is that they parse the input questions to determine the syntactic structure of the query before deriving a deeper meaning representation or using the syntactic analysis for another subprocess.

Despite this heavy dependence on parser-based shallow and deep linguistic analysis of questions for QA, surprisingly little research has been carried out on how well state-of-the-art shallow and deep parsing systems cope with question analysis. To the best of my knowledge, no research has been carried out to establish whether high performing Penn-II Treebank trained (Marcus et al., 1993, 1994) modern parsers like Collins (1999), Charniak (2000) and Bikel (2002) or deeper linguistic analysis tools, for example the automatic f-structure annotation algorithm of Cahill et al. (2004), are able to maintain their high accuracy when tested on questions, instead of on Wall Street Journal text from the Penn-II Treebank (Marcus et al., 1994).

This thesis presents research on examining how state-of-the-art parsing and f-structure annotation systems cope with question material, the adaption of such resources to optimally cope with questions, and the creation of resources to support research and development in parser-based analysis of questions.

Probabilistic parsing resources reflect the characteristics of their training material and generally underperform on material which differs from the training material. This mismatch between training and test/evaluation material is referred to as domain variation.

The research I present investigates strong domain variation, and its effects on parser performance. Parsing questions is an instance of domain variation, but, as I will show, it constitutes an instance of more severe domain variation than was observed in previous studies (Gildea, 2001), and as a result the effects are much more pronounced. I show that the ATIS corpus (Hemphill et al., 1990) is substantially different from the Penn-II Treebank and that it contains a high proportion of questions. This makes it useful for an initial investigation into domain variance in a question-rich domain.

I test the Penn-II trained parsers of Collins (1999), Charniak (2000) and Bikel (2002) on ATIS data and show that each of the parsers suffers a drop in performance in the new domain with an average drop in labelled precision and recall f-score of 19.99% when compared to the results for parsing Section 23 of the Penn-II Treebank. Following Gildea (2001) I add appropriate data to the parsers' training corpus to allow them to cope with the new domain. This boosts the parsers' performance on the ATIS data significantly. Bikel's result on a held out test set of 10% of the ATIS corpus increases from an f-score of 64.84% to 85.20% and Charniak's increases from 63.64% to 81.59%.[1]

Following from this, I present the first domain variance research which investigates the effects of domain variance on the Penn-II Treebank- and f-structure annotation algorithm-based LFG parsing resources of Cahill et al. (2004). I show that the preds-only dependency f-score in a test on a 100 hand-crafted f-structure gold standard for ATIS sentences drops to 62.95%, compared to 74.10% for the Penn-II based DCU 105 gold standard. Due to the modular, pipeline design of the LFG parsing architecture of Cahill et al. (2004), the observed underperformance could be the result of one or several stages in the parsing architecture being sub-optimal for question analysis. I show that the underperformance stems from the c-structure parser and that retraining the parser significantly improves the quality of both c- and f-structure analysis increasing the preds-only f-score by 13.82% to 76.77% in an evaluation on the held out 100 ATIS sentence gold standard.

The work shows that the automatic f-structure annotation algorithm of Cahill et al. (2004) is complete with respect to the domain variance observed in this thesis, as it did not need to be modified to cope with the new domain. It also supports the observation that f-structures are a more abstract representation of linguistic information, less affected by domain variation.

The retraining work presented in the first two chapters of the dissertation highlights the benefit that retraining a parser on a small amount of domain appropriate data can have in both c- and f-structure analysis of out-of-domain data. The ATIS corpus, however, is quite small, not representative of a large amount of question types and contains extraneous non-question data. So, while it is useful for initial investigations, these properties make the ATIS resource unsuitable for larger evaluations and for use as a fully fledged question corpus. To address this, I have semi-automatically created a new training and development resource for parser-based linguistic analysis of questions. QuestionBank is a parse-annotated corpus of 4000 questions following Penn-II guidelines (Bies et al., 1995), intended for use as a training and evaluation corpus for parsing-based technology used in question answering. I have used QuestionBank

[1] Collins' parser does not come with the functionality to retrain it on new training data, so I am unable to give results for this parser in the retraining experiments.

to retrain Bikel's parser to be able to parse questions with a high degree of accuracy (88.82% labelled precision and recall f-score) and, when trained in conjunction with Sections 02-21 of the Penn-II Treebank, the parser can not only parse questions but also informative text in Section 23 of the Penn-II Treebank with state-of-the-art accuracy.

Long Distance Dependencies (LDDs) are important in the analysis of English wh-questions, as the wh-phrase of the question generally refers to a dislocated element corresponding to the answer to the question. Because of this, correctly identifying LDDs in input questions significantly improves the quality of question analysis. However, most state-of-the-art probabilistic parsers do not include this kind of information in their output.[2] I have developed a novel method for recovering traces in parser output which uses reentrancies in automatically generated long distance dependency resolved f-structures to "reverse engineer" the corresponding c-structure trace and coindexation. This method proved quite successful in evaluations on questions from QuestionBank with an f-score of 80.78% on gold standard trees stripped of traces and functional information, and 68.99% on parser output. When compared with other systems available for the task, the results show that this method for recovering LDD information outperforms the others.

This thesis is structured as follows:

Chapter 2 introduces Question Answering, parsing, Lexical Functional Grammar, shows how linguistic analysis is useful to QA, sets the context and gives some background for the research presented in this thesis.

Chapter 3 contrasts the ATIS corpus and Penn-II Treebank and presents preliminary research on parsing ATIS data with state-of-the-art parsers and shows how the parsers can be adapted to better cope with ATIS data.

Chapter 4 examines domain variance in automatic f-structure annotation and shows that the observed performance drop is due to the c-structure parser underperformance on the out-of-domain data, which results in poor quality f-structure annotation.

[2] Collins' Model 3 parser is a notable exception which outputs traces for wh-relative clauses.

Chapter 5 describes the bootstrapping of QuestionBank, a parse annotated corpus of 4000 questions, intended to function as a development resource for parser-based linguistic tools for use in QA. QuestionBank was created semi-automatically from raw data taken from state-of-the-art development and test sets for QA. This chapter describes the raw data, and the semi-automatic "bootstrapping" method used to create QuestionBank.

Chapter 6 presents a series of experiments with QuestionBank to determine its effectiveness as a training resource for question parsing and to investigate the effect that adapting a parser to accurately analyse questions (out-of-domain data) has on its ability to parse informative text (in-domain data).

Chapter 7 describes a novel method to recover trace information in parser output using automatically generated, long distance dependency resolved f-structures to recreate the corresponding trace and coindexation information in the tree. This method is used to induce trace information in QuestionBank and compared with the approaches of Johnson (2002) and Higgins (2003).

Chapter 8 concludes and outlines some areas of future work.

Much of the experimental work in this thesis addresses a space of possibilities that arise from training and testing parsers on question-rich versus statement-rich corpora. The general space of possibilities can be represented by the following table:

Test Training	C/F-structure Evaluation on PTB	C/F-structure Evaluation on Question-rich Data
PTB		
Question-rich		
Question-rich + PTB		

A number of chapters address areas within this space of possibilities. I will revisit this table (extending it where necessary) to illustrate the area(s) applicable to particular chapters.

Chapter 2

Background

2.1 Introduction

This chapter sets the context and background relevant to the research presented in this dissertation. I introduce Question Answering (QA), an area within Information Retrieval (IR), and discuss newly emerging directions in QA. I provide an overview of shallow and deep natural language parsing methods (based on Context Free Phrase-Structure Grammars and Lexical Functional Grammar) and present treebank-based automatic deep grammar acquisition based on the automatic f-structure annotation algorithm of Cahill et al. (2004). I present a short overview of the Penn-II Treebank data structures and encoding of syntactic information relevant to this dissertation. I review research on combining NLP methods with Question Answering, and describe how improving and adapting NLP tools for questions can benefit the Question Answering process.

2.2 Question Answering

Information Retrieval (IR) is the process of finding information in a data repository in response to a user query.[1] The information repository can consist of structured data, like a database,

[1] In this thesis I restrict myself to text-based IR.

or unstructured data, like a collection of documents or the worldwide web (WWW). The most prevalent information retrieval systems are search engines for the worldwide web. Search engines index very large amounts of data from webpages and in the simplest case use keyword matching techniques to retrieve documents relevant to a query. In recent years internet search engines have started to employ more sophisticated techniques which take advantage of the nature of hypertext in the WWW, including for example linkage analysis (Brin and Page, 1998), as well as shallow linguistic techniques, including, for example, query expansion via synonyms, to improve document retrieval and ranking. Despite these advances, it is often the case that the highest ranked documents do not contain the information the user needs (Silverstein et al., 1998), leaving the user to search a (quite large) collection of potentially relevant documents for the information queried, or abandon his/her search if this proves too time consuming. As the amount of information available on-line grows, this problem becomes further compounded and the need for systems which can deliver precise information in response to a precise query becomes more urgent.

Question Answering (QA) addresses this need by combining linguistic processing with "standard" IR technology in a way that allows the user to state his/her information need as s/he would naturally, i.e. in the form of a question, and to receive a concise response, rather than complete documents, containing the information (or a ranked list of candidate responses) which satisfies the information need, i.e. answers the question.

The QA paradigm is regarded as a refinement of standard document retrieval-based IR in that it is narrowing the scope of both the query and the response to a specific information need, a question, and, for simple fact seeking (factoid) questions, a specific response to that information need, a fact. With this in mind it is clear that QA is an augmentation to, but not a replacement for, IR.

Information Retrieval has been actively researched since as far back as the mid-1950's. Its relationship to Question Answering is that users form queries because they require information to find the answers to questions. Beyond this, the similarities largely end. IR systems return documents instead of answers, from which users must extract the information themselves. The documents are generally ranked on the basis of keyword matching heuristics and linkage analysis and do not include other specific constraints set out in the query. Queries put to IR systems are often treated as a collection of keywords and a query is not required to be well formed syntactically.

This "bag of words" approach can be a drawback as subtle but important differences between queries can be lost. Consider, for example, the following two queries

1. "Who killed Harvey Oswald?"

2. "Who did Harvey Oswald kill?"

In the first case, the query is for the agent/subject of the observed eventuality (i.e. Jack Ruby); in the second, it is for the patient/object of the (different) described eventuality (i.e. JFK). This important difference is, however, lost on standard IR systems, which reduce both inputs to the same set of stemmed open class query terms. After stopword removal the queries yield:

$$\{\texttt{killed,Harvey,Oswald}\}\ \{\texttt{Harvey,Oswald,kill}\}$$

After stemming this is reduced to equivalent sets of query terms, generating identical responses from the IR system:

$$\{\texttt{kill,Harvey,Oswald}\}\ \{\texttt{Harvey,Oswald,kill}\}$$

In order to capture important differences such as these, a QA system must perform deeper linguistic analysis than stemming and stopword removal. In the case at hand, the QA system must determine the syntactic/semantic roles of the participants in the described eventuality. This involves resolving the wh-pronoun in the first query as subject/agent and as object/patient in the second. Notice that in general this may involve long distance dependency resolution and effectively amounts to the construction of predicate-argument representations.

2.2.1 History of QA

Although information systems and IR as we know it today did not really take off until the 1990's with the advent of the PC, earlier forms of IR and QA have been around for quite some time. One of the earliest surveys on QA is a paper published in 1965 (Simmons, 1965) which reviews fifteen QA systems that had been developed over the previous 5 years. Simmons categorises these systems into 5 types: list-structured, data-based, graphic data-based, text-based and inferential. The first three correspond to systems which rely on a structured database of one form or other. The text-based systems represent the beginnings of modern IR systems, using collections of documents instead of structured databases to answer questions. The inferential systems described in Simmons' paper translate questions into a quasi-logical form and infer answers to the questions based on a database of logical facts, an approach that is still used (at least in part) by some modern QA systems (Waldinger et al., 2004).

Given the state-of-the-art in document collection and processing at the time, it is no surprise that the more successful of the early QA systems were those designed as a front-end to a structured database. One of the most successful systems in this category (developed after Simmons' article) accessed a database of geological information on rock samples brought back from the moon during the Apollo missions. The system, LUNAR (Woods, 1973), was capable of answering 90% of the in-domain questions posed to it. This encouraging early result spawned a plethora of important research in the database front-end approach, including the BASEBALL and PLANES systems (Green et al., 1961; Grosz et al., 1986) through the '70s and '80s.

The early database front-end approach, though successful, is limited in that it is restricted to a closed information domain and relies on having the information repository stored as a structured database. Recent advances in IR have produced very good results in searching over large document collections like WT10G,[2] .GOV[3] and the WWW, containing unstructured and diverse information in text documents. QA research has followed suit, moving from closed to open

[2] http://es.csiro.au/TRECWeb/wt10ginfo.ps.gz

[3] http://ir.dcs.gla.ac.uk/test_collections/govinfo.html

domain questions and unstructured, text-based datasets. Expanding from a small closed domain into open domain free text QA is not as simple as just expanding the dataset and test set. The QA system itself has to change and "grow" to accommodate the expanded requirements (Hirschman and Gaizauskas, 2001). Input to an open-domain QA system can come from any subject area. The system needs to expand its language coverage to be able to handle unrestricted natural language queries, and to be able to process unrestricted texts to find answers.

Information Extraction (IE) research has in many ways fed into current QA research. IE has been described as using natural language texts to fill data templates which represent stereotypical events (Hirschman and Gaizauskas, 2001). For example, an IE system may involve templates to extract information about people's births and deaths from newspaper obituaries. An IE template can be loosely compared to a question and a filled template can be regarded as containing an answer. The IE community's competitive evaluations at the Message Understanding Conferences (MUCs) ran from 1987 to 1998. In most current research, IE has become an integral part of modern IR and QA systems.

In 1999 the Text Retrieval Conferences (TREC) organised by the U.S. National Institute for Science and Technology (NIST) introduced a QA track (Voorhees, 2001).[4] The purpose of this track is to promote research in language understanding technologies through QA and to provide an evaluation forum for such research. The TREC evaluations have now become a benchmark for English QA evaluations. The questions used in the evaluations of open domain QA systems tend to be short fact seeking (factoid) questions, which usually require a named entity, or an amount, location etc., to respond to the query.

2.2.2 New Directions in QA

After over 40 years of R&D in QA, research is now beginning to fully explore the area and realise its potential in terms of prototype systems and products. Even the limited solutions offered by current systems provide added value over course-grained document-based IR. The range and

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

scale of systems employed in the TREC evaluations indicate that QA is an area of growth and diversity drawing on recent advances in both IR and NLP research.

In recent years, cross-language retrieval tasks have emerged, with several conferences and workshops (CLEF, NTCIR) dedicated to the area. Cross-Language Information Retrieval (CLIR) and Cross-Language Question Answering (CLQA) are similar to their monolingual counterparts except that the queries are in one language and the document collection in another. The ability to utilise documents from other languages is useful to QA because it enables a system to draw on information in a language different to the query. By allowing a system to search a (much) larger document collection, the chances of finding a correct answer are increased. There are, however, drawbacks to increasing the size of the document collection with similar documents from other languages. There is an added cost in terms of processing and managing the document collection, and there are also problems associated with noise introduced by similar but irrelevant texts in the document collection.

Given that CLQA/IR systems have already started to bridge the language gap to retrieve appropriate documents and answers from different languages, and given the recent advances in Multimedia IR (MMIR) research (Smeaton et al., 2002), another logical progression for QA research is Multimedia QA (MMQA). A MMQA system draws on more than text documents to derive the answers it returns, looking also at the information contained in images, video, and potentially speech. This is an exciting new area in QA research, offering the potential to not only answer a user's question but also to return appropriate images, video clips and sound bytes to back up the answer or to provide more relevant information to the user.

The technique of asking questions about a text to test whether a person has understood it correctly has been successfully employed in teaching both children and second language learners. Hirschman et al. (1999) describe a system which takes a children's story as input and can answer questions on the text. This type of QA system presents new and interesting uses in the field of Computer-Aided Language Learning (CALL) and, more generally, E-Learning. Such a QA system could conceivably be used to grade students' answers to questions about unseen text.

Alternatively, in a wider learning context a QA system could be part of an automated companion learning program which "learns" at the same time as the student.

2.3 Shallow and Deep NLP Methods

In order to effectively return answers to questions, a QA system needs to perform some level of linguistic analysis on the input questions. This allows the QA system to capture important differences between queries, e.g. the agent/patient (subject/object) difference between "Who killed Harvey Oswald?" and "Who did Harvey Oswald kill?" discussed in examples (1) and (2). The level and nature of linguistic processing employed by a QA system varies depending on the architecture of the system. In this section, I will outline a shallow analysis method, Context Free Grammar-Based Parsing, which has been applied in query/document analysis for high performance QA (Pasca and Harabagiu, 2001). I will then outline a deep analysis method, Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982), which analyses sentences into basic predicate-argument structures with long distance dependency relations resolved, and give a brief overview of some work which has been done on parsing raw text into LFG f-structures.

2.3.1 Context-Free Grammar (CFG) Parsing

Parsing is the process of analysing (natural or formal) language strings into their component parts and describing how they relate to each other syntactically. For natural language input, CFG parsing usually produces an output showing the lexical category of each of the words (its part-of-speech, POS) and the internal structure of the sentence (a parse tree). CFG parsing is a useful step in processing natural language text as a syntactic analysis guides the semantic interpretation of an input string in the form of dependencies, predicate-argument structures, or logical forms.

CFG parsing assigns syntactic structure to a string according to the rules provided by a grammar for the language. A grammar is a set of rules which describe what is a valid construction for the language (fragment) described by the grammar. A Context Free Grammar (CFG) is a grammar in which every production rule is of the form

$$\mathrm{A} \rightarrow \alpha$$

where A is a non-terminal and α is a set of terminals and/or non-terminals. Context free grammars are called "context free" because A can always be replaced by α regardless of the context in which it occurs. Context free grammars define a class 2 language according to the Chomsky hierarchy (Figure 2.1)

Language class	Grammar	Automaton
3	Regular	NFA or DFA
2	Context-Free	Push-Down Automaton
1	Context-Sensitive	Linear-Bounded Automaton
0	Free (Unrestricted)	Turing Machine

Figure 2.1: The Chomsky hierarchy of languages, grammars and automata

Formally, a CFG G can be described as a 4-tuple $G = \langle V_t, V_n, P, S\rangle$, such that

- V_t is a finite, non-empty set of terminals
- V_n is a finite, non-empty set of non-terminals
- P is a finite, non-empty set of production rules of the form $V_n \rightarrow (V_t \cup V_n)^*$
- $S \in V_n$ is the distinguished start symbol

Modern CFG parsers for NLP applications generally do not use simple CFG grammars to parse strings. This is because in cases where multiple analyses can be assigned to an input string, a CFG cannot output a "best" parse for the input string, but can only enumerate each of the possible parses. A simple CFG does not distinguish preferred (commonly used) constructions from rare ones. In order to do this, CFG parsers such as LoPar (Schmid, 2000) use Probabilistic Context-Free Grammars (PCFGs), an extension of CFGs, which associate a probability with each production rule A $\longrightarrow \alpha$. Formally, a PCFG G can be described as a 5-tuple $G = \langle V_t, V_n, P, S, R \rangle$, such that

- V_t, V_n, P and S defined as for a CFG
- R is a function which assigns a probability to each rule A $\longrightarrow \alpha \in P$ such that for each LHS: $\Sigma_{LHS} R(RHS \rightarrow LHS) = 1$

Using this model, a PCFG defines the probability of a parse tree T given a string S, i.e. P(T|S), as the product of the probabilities of each of the productions in T. A PCFG parser can then choose the most likely analysis for a string S as the parse tree T which maximises P(T|S).

The probability associated with each of the rules in a PCFG has a great effect on how that rule influences a parse tree derivation. Often both the grammar rules and the rule probabilities are extracted from a parse-annotated treebank and each rule's probability is estimated in terms of its relative frequency in the treebank.[5]

$$P(LHS \rightarrow RHS_j) = \frac{\#(LHS \rightarrow RHS_j)}{\sum_{i=1}^{n} \#(LHS \rightarrow RHS_i)} \tag{2.1}$$

Given a suitably large treebank, PCFG parsing can achieve very high coverage and is able to rank output alternatives. Everything else being equal, PCFG-based parsing favours derivations with small numbers of expansions, and hence a small number of probabilities to multiply out, which results in a bias towards smaller, less hierarchical trees with less structure.

Other state-of-the-art approaches to probabilistic, wide coverage parsing use more sophisticated mechanisms than the simple PCFG model outlined above to produce CFG parse trees. An important family of state-of-the-art parsers producing CFG trees use richer language models

[5] Here # stands for counts.

(with fewer independence assumptions) than a PCFG. The parsers of Collins (1999), Charniak (2000) and Bikel (2002) employ history-based, generative, lexicalised models and achieve results of almost 90% labelled f-score when tested on the trees in Section 23 of the Penn-II Treebank. These parsers are the basis of a number of my experiments described in Chapters 3, 4 and 6.

Collins (1999) introduces three parsing models (1, 2 and 3) for a history-based parser (Black et al., 1993). Collins' history-based parsing model is defined in terms of a top-down leftmost derivation where (in principle) anything previously generated by the derivation process can appear in the conditioning context for the expansion of the next non-terminal. Model 1 is a basic history-based model which tries to overcome the sparse data problem of lexicalised parsing by first generating the head of a constituent, followed by its left and right contexts. Model 2 distinguishes complements and adjuncts and Model 3 can produce traces for wh-relative clause movement. Training on Sections 02-21 and evaluating on Section 23 of the Penn-II Treebank (Collins, 1999) shows that Collins' Model 3 achieves the highest results with precision of 88.7% and recall of 88.6% on sentences of length ≤ 40 and Model 2 outperforms Model 1 with precision of 88.7% and recall of 88.5% on sentences of length ≤ 40. For each model, the results are slightly worse for sentences of length ≤ 100. Table 2.1 summarises the results.

	Model 1		Model 2		Model 3	
	LP	LR	LP	LR	LP	LR
$\leq$ 40 words	88.2	87.9	88.7	88.5	88.7	88.6
$\leq$ 100 words	87.7	87.5	88.3	88.1	88.3	88.0

Table 2.1: Parsing results for Collins' parser on Section 23 of the Penn-II Treebank

The parser described in Charniak (2000) is based on a probabilistic generative model (Charniak, 1997), which for a sentence S and parse tree T assigns the probability P(S,T) = P(T). The parser returns the tree which maximises this probability. A probability is assigned to a tree T by a top-down process considering each constituent in the tree and assigning a probability based on the constituent's label, lexical head and generative history. Training this parser on Sections 02-21 of the Penn-II Treebank and testing on Section 23, Charniak's parser outperforms Collins', achieving labelled precision and recall of 90.1 on sentences of length $\leq$ 40 with only a slight performance drop for sentences of length $\leq$ 100. Table 2.2 (Charniak, 2000) summarises these results.

	Labelled Precision	Labelled Recall
$\leq$ 40 words	90.1	90.1
$\leq$ 100 words	89.6	89.5

Table 2.2: Parsing results for Charniak's parser on Section 23 of the Penn-II Treebank

Bikel (2002) provides a Java implementation of a retrainable and extensible multilingual parser which emulates Collins (1999) Model 2. The parsing engine provides a number of language packages and an extensive API to develop packages for new languages. Table 2.3 shows parsing results for Collins' original Model 2 and Bikel's emulation on Section 00 of the Penn-II Treebank.

	Collins Model 2		Bikel	
	LP	LR	LP	LR
$\leq$ 40 words	89.75	90.19	89.89	90.14
$\leq$ 100 words	88.47	89.30	88.72	89.03

Table 2.3: Parsing results for Collins' Model 2 parser and Bikel's emulation on Section 00 of the Penn Treebank

2.3.2 Lexical Functional Grammar

Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001) is a constraint-based grammar formalism. LFG (minimally) involves two levels of representation, c(onstituent)-structure and f(unctional) -structure. C-structure takes the form of CFG trees which capture language specific issues like constituent grouping and word order. F-structure represents a deeper, more abstract level of linguistic information such as subject, object, complement etc., in the form of Attribute Value Matrices (AVMs) which approximate to basic predicate-argument structures or deep dependency relations.

C-structures are related to f-structures through ϕ-projections which map c-structure nodes to their corresponding f-structures. Functional annotations on the c-structure nodes representing constraints which describe the f-structure. Figure 2.2[6] shows an example c- and f-structure for the sentence "John saw Mary" where the ϕ-correspondence between the c-structure tree and the f-structure is indicated in terms of arrows from c-structure nodes to f-structure nodes.

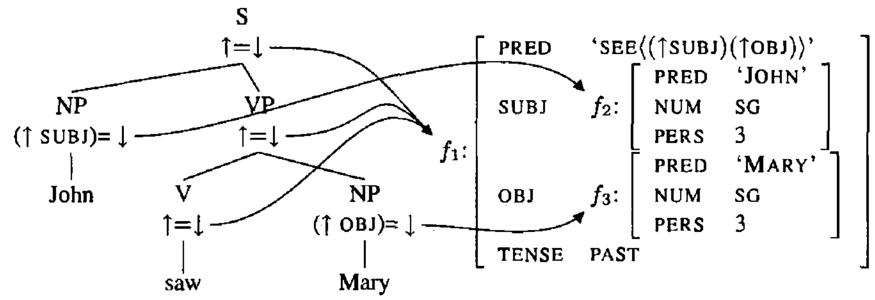


Figure 2.2: C- and f-structures for the sentence "John saw Mary"

The up- and down-arrows in the functional annotation (↑↓) refer to the f-structure associated with the mother node (↑) and the local node (↓). These are instantiated to unique tree node identifiers and, if all the constraints are satisfiable, an f-structure is *generated* from the annotated tree.

[6] Lexical annotations have been suppressed to aid readablity

Wellformedness Conditions

F-structures are required to meet three wellformedness conditions: Completeness, Coherence and Uniqueness (Kaplan and Bresnan, 1982). These conditions ensure that a predicate has all of the arguments which it requires, that there are no additional arguments and that each attribute has a single value.

The completeness condition states that an f-structure is *locally complete* if and only if it contains all of the governable grammatical functions that its predicate governs, and an f-structure is *complete* if and only if all of its sub-f-structures are locally complete (Kaplan and Bresnan, 1982). This condition ensures that the f-structure for a sentence like

*John threw.

$$\begin{bmatrix} \text{SUBJ} & \text{`John'} \\ \text{PRED} & \text{``THROW}\langle\text{SUBJ,OBJ}\rangle\text{''} \end{bmatrix}$$

is incomplete because some required material is missing. The main verb of the sentence is transitive and requires both a subject and an object. Its semantic form subcategorises for subject and object arguments:

$$\begin{bmatrix} \text{PRED} & \text{``THROW}\langle\text{SUBJ,OBJ}\rangle\text{''} \end{bmatrix}$$

In the f-structure above only the SUBJ role governed by the local PRED is present, therefore the f-structure for the whole sentence is incomplete.

The coherence condition disallows f-structures which contain extra governable grammatical functions which are not governed by the local predicate. Formally, an f-structure is *locally coherent* if and only if all of the governable grammatical functions it contains are governed by a local predicate. An f-structure is *coherent* if and only if all of its sub-f-structures are coherent (Kaplan and Bresnan, 1982). This ensures that a sentence like:

*John slept the ball.

$$\begin{bmatrix} \text{SUBJ} & \text{`John'} \\ \text{PRED} & \text{``SLEEP}\langle\text{SUBJ}\rangle\text{''} \\ \text{OBJ} & \text{`the ball'} \end{bmatrix}$$

is incoherent because the object is not governed by the local intransitive verb “sleep” which only subcategorises for a SUBJ as in the semantic form below.

$$\begin{bmatrix} \text{PRED} & \text{“SLEEP}\langle\text{SUBJ}\rangle\text{”} \end{bmatrix}$$

The uniqueness condition requires that each attribute of an f-structure has at most one value. This prevents f-structures from having incompatible constraints, for example:

*The boys sleeps.

$$\begin{bmatrix} \text{PRED} & \text{“SLEEP}\langle\text{SUBJ}\rangle\text{”} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘BOYS’} \\ \text{NUM} & \text{SG/PL} \end{bmatrix} \end{bmatrix}$$

The subject noun phrase “the boys” is plural, but the verb “sleeps” requires a singular subject. Since the NUM value cannot be both singular *and* plural there is a feature clash violating the uniqueness condition so the f-structure is not well formed.

LFG is an interesting framework because f-structure abstracts away from some language specific issues like word order, associated with the surface string, approximating to predicate-argument structure, deep dependencies or a simple logical form. Figure 2.3 shows an example English sentence and its corresponding translation in Irish. Note that despite the different word order (English is an SVO language, Irish is a VSO language) the f-structures are isomorphic (up to leaf node relabelling). LFG is an attractive formalism for question analysis because long-distance dependencies are resolved in f-structures providing important information about both informative text and questions (this is the topic of Chapter 7 of this thesis).

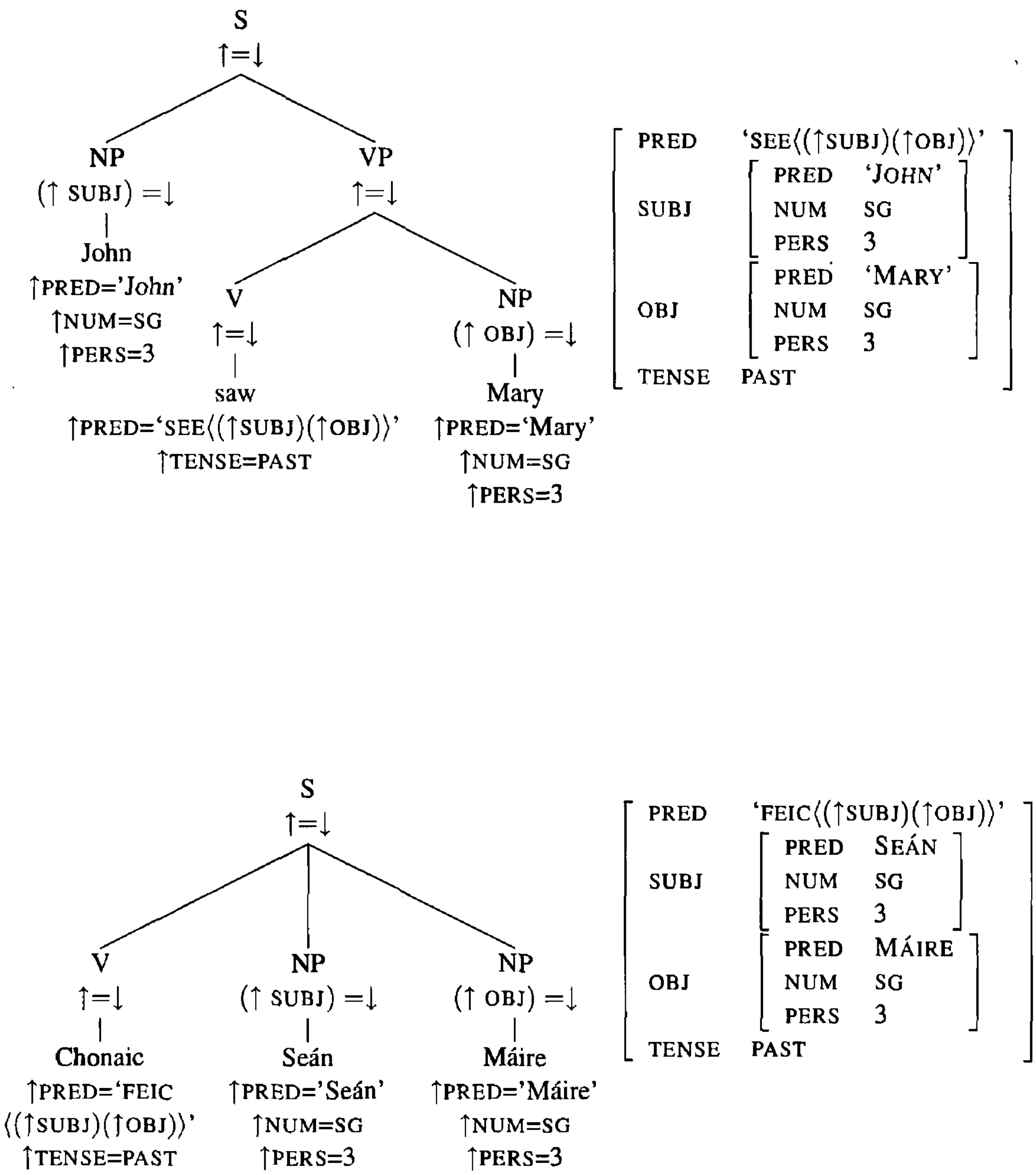


Figure 2.3: C- and f-structures for an English and corresponding Irish sentence.

2.3.3 Treebank-Based Acquisition of LFG Resources and "Deep" Parsing Using LFG

LFG (and similar constraint-based grammar formalisms like HPSG, CCG and TAG) represents "deep" linguistic information in terms of grammatical function and dependencies. However, hand-crafting deep constraint grammar resources and scaling them to unrestricted text is prohibitively time consuming and expensive. Because of this, a number of researchers have developed automatic treebank-based acquisition methods for constraint-based grammars (Cahill et al. (2002a, 2004), Miyao et al. (2003), Hockenmaier (2003a)). In this section, I give an outline of the Automatic LFG F-structure Annotation Algorithm of Cahill et al. (2002a, 2004) and briefly describe the two parsing architectures presented in Cahill et al. (2002b): *the pipeline model*, which takes CFG parse trees and adds LFG annotations to generate f-structures, and *the integrated model*, which uses an f-structure Annotated PCFG (APCFG). I use the automatic f-structure annotation algorithm in experiments on the ATIS corpus (Hemphill et al., 1990) in Chapter 4 and I use APCFGs in the reconstruction of LDDs in a bootstrapped question treebank in Chapter 7.

Automatic f-structure annotation has previously been explored on a much smaller scale by Lappin et al. (1989), Sadler et al. (2000) and Frank (2000). The first large-scale automatic f-structure annotation project was Cahill et al. (2002a) which uses an annotation algorithm to automatically annotate Penn-II Treebank-style CFG trees with functional equations.

The annotation algorithm is used to annotate treebank trees and parser output trees. Penn-II treebank trees encode long-distance dependencies (LDDs) in terms of empty productions (traces) and coindexation in trees. A module (traces) in the annotation annotation algorithm (Figure 2.4) translates these into corresponding reentrancies to represent the LDD at f-structure. Probabilistic parser output trees do not generally represent LDDs (they do not produce traces and coindexations). A separate LDD-resolution module (Cahill et al., 2004) resolves LDDs at f-structure for parser output (Figure 2.9). Below, I will first present the f-structure annotation algorithm as it applies to treebank trees and then describe how the annotation algorithm is integrated into the parsing architectures of Cahill et al. (2004).

The annotation algorithm is embedded in a 2 stage process (Figure 2.4): the treebank trees are annotated with functional equations by the annotation algorithm and then passed to a constraint solver to generate f-structures.

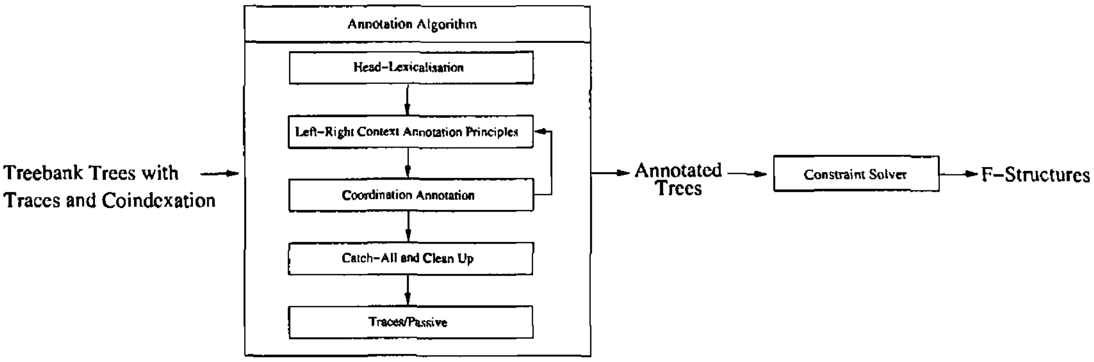


Figure 2.4: Automatic F-Structure annotation Algorithm of Cahill et al. (2004)

The annotation algorithm consists of 5 sub-modules:

Head Lexicalisation The algorithm looks at each local subtree of depth 1 and uses a modified version of Magerman's (1994) head-finding rules to partition the daughters of the subtree into a head (h), left context daughters ($l_1 \cdots l_n$) and right context daughters ($r_1 \cdots r_m$): MOTHER $\rightarrow l_1 \cdots l_n \; h \; r_1 \cdots r_m$.

Left-Right Context Annotation Based on the partition derived in the Head Lexicalisation module, the algorithm uses categorial and configurational information to annotate each of the daughters. For example a determiner to the left of the head of an NP gets the annotation $\uparrow$ SPEC:DET $=\downarrow$.

Coordination Coordinate structures in the Penn-II Treebank are flat and can be difficult to analyse. Because of this, coordinations are treated separately in order to keep the left-right context annotation principles simple and perspicuous. For like- and unlike- constituent coordinations the algorithm uses coordination sets. Using these sets the algorithm decides which daughters form part of a (local) coordination and to annotate them accordingly and which remaining daughters are annotated by the regular left-right annotation principles.

Catch-All and Clean Up This module is responsible for correcting over-generalisations that arise from the previous annotation modules. Default annotations on some nodes are overwritten using rules and information from the Penn-II functional tags (-CLR, -DTV, etc.) to catch specific overgeneralisations.

Traces Long Distance Dependencies are encoded in the Penn-II Treebank by means of empty nodes and trace coindexation between the empty node and the dislocated element. The trace module links the trace nodes with their associated antecedent in terms of a corresponding reentrancy in the f-structure. Annotating passive constructions is also carried out by this module.

The automatic LFG annotation algorithm takes Penn-II Treebank trees as input and outputs f-structures for the trees. Figure 2.5 shows example input to and output produced by the annotation algorithm (Figure 2.4) for the sentence "Who did Mary see?".

Before annotation:

```
(SBARQ (WHNP-1 (WP Who)) (SQ (AUX did) (NP (NNP Mary)) (VP (VB see) (NP
(-NONE- *T*-1)) )) (.  ?))
```

After annotation:

```
(SBARQ (WHNP-1[up-focus=down] (WP[up=down] Who[up-pred=pro,up-pron_form='who']))
(SQ[up=down] (AUX[up=down] did[up-pred='did'])(NP[up-subj=down]
(NNP[up=down] Mary[up-pred='mary',up-num=sg,up-pers=3]))(VP [up-xcomp=down,
up-subj=down:subj] (VB[up=down] see[up-pred='see'])(NP[up-obj=down,
F1:focus===Fdown, up-resolved=focus] (-NONE- *T*-1))))(.  ?))
```

Figure 2.5: Example input and output from the annotation algorithm of Cahill et al. (2004).

The annotations are then converted to a PROLOG representation similar to Figure 2.6 (a) and passed to the constraint solver, which creates f-structures for the input sentences if the equations are satisfiable (Figure 2.6).

(a)

```
fstruct(bnc_1, F1):-
F1:focus===F2,
F2===F3,
F3:pred===pro,
F3:pron_form==='who',
F1===F5,
F5===F6,
F6:pred==='do',
F5:subj===F8,
F8===F9,
F9:pred==='mary',
F9:num===sg,
F9:pers===3,
F5:xcomp===F11,
F5:subj===F11:subj,
F11===F12,
F12:pred==='see',
F11:obj===F14,
F1:focus===F14,
F11:resolved===focus.
```

(b)

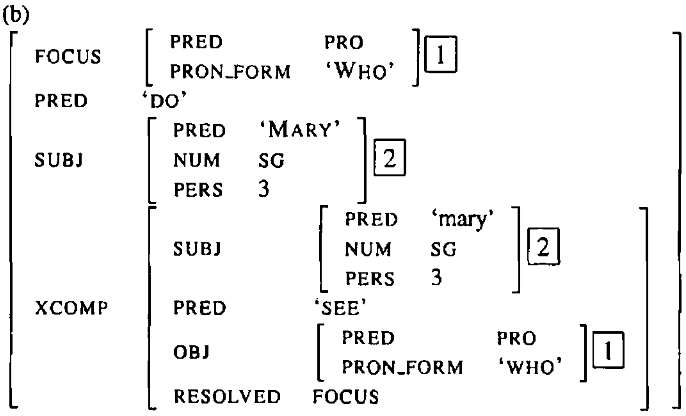


Figure 2.6: PROLOG format input to the constraint solver (a) and the human readable f-structure output (b).

Note that the LDD in the input tree is represented by a reentrancy between the FOCUS and OBJ functions in the f-structure in Figure 2.6 (b) which is indicated by the indices.

If the treebank trees input to the annotation algorithm indicate long distance dependencies in terms of traces and coindexation (Figure 2.7) then the f-structures generated by the annotation algorithm and the constraint solver indicates LDDs in terms of corresponding reentrancies at f-structure. This is because the Trace module of the annotation algorithm (Figure 2.4) generates annotations from empty nodes and their antecedents which define corresponding reentrancies in the f-structures.

Parser Output

```
(SBARQ (WHNP (WP Who)) (SQ (AUX did) (NP (NNP Mary)) (VP (VB see))) (. ?))
```

Annotated Tree

```
(SBARQ (WHNP[up-focus=down] (WP[up=down] Who[up-pred=pro,up-pron_form='who']))
(SQ[up=down] (AUX[up=down] did[up-pred='did'])(NP[up-subj=down] (NNP[up=down]
Mary[up-pred='mary',up-num=sg,up-pers=3]))(VP[up-xcomp=down,up-subj=down:subj]
(VB[up=down] see[up-pred='see'])))(. ?))
```

Figure 2.7: Parser output tree, and annotated parser output for “Who did Mary see?”

However, if the input trees do not have long distance dependencies resolved, for example in parser output (Figure 2.7), then the f-structures output by the constraint solver will not be LDD resolved. In this case a further processing step is necessary. The unresolved (proto) f-structures output by the constraint solver are then passed to the f-structure LDD resolution module described in Cahill et al. (2004) which resolves long distance dependencies at f-structure level. This LDD resolution module uses reentrancy paths (finite approximations of functional uncertainty equations) and verb subcategorisation frame information learned from f-structures automatically generated from the Penn-II Treebank to LDD resolve the unresolved f-structures. Figure 2.8 shows the f-structure generated from the parser output (unresolved) tree for “Who saw Mary?” before and after the f-structure LDD resolution module is applied.

(a)

$$\left[\begin{array}{ll} \text{FOCUS} & \left[\begin{array}{ll}\text{PRED} & \text{PRO}\\ \text{PRON\_FORM} & \text{`WHO'}\end{array}\right]\\ \text{PRED} & \text{`DO'}\\ \text{SUBJ} & \left[\begin{array}{ll}\text{PRED} & \text{`MARY'}\\ \text{NUM} & \text{SG}\\ \text{PERS} & 3\end{array}\right]\\ \text{XCOMP} & \left[\begin{array}{ll}\text{SUBJ} & \left[\begin{array}{ll}\text{PRED} & \text{`MARY'}\\ \text{NUM} & \text{SG}\\ \text{PERS} & 3\end{array}\right]\\ \text{PRED} & \text{`SEE'}\end{array}\right]\end{array}\right]$$

(b)

$$\left[\begin{array}{ll} \text{FOCUS} & \left[\begin{array}{ll}\text{PRED} & \text{PRO}\\ \text{PRON\_FORM} & \text{`WHO'}\end{array}\right]\boxed{1}\\ \text{PRED} & \text{`DO'}\\ \text{SUBJ} & \left[\begin{array}{ll}\text{PRED} & \text{`MARY'}\\ \text{NUM} & \text{SG}\\ \text{PERS} & 3\end{array}\right]\boxed{2}\\ \text{XCOMP} & \left[\begin{array}{ll}\text{SUBJ} & \left[\begin{array}{ll}\text{PRED} & \text{`MARY'}\\ \text{NUM} & \text{SG}\\ \text{PERS} & 3\end{array}\right]\boxed{2}\\ \text{PRED} & \text{`SEE'}\\ \text{OBJ} & \left[\begin{array}{ll}\text{PRED} & \text{PRO}\\ \text{PRON\_FORM} & \text{`WHO'}\end{array}\right]\boxed{1}\\ \text{RESOLVED} & \text{FOCUS}\end{array}\right]\end{array}\right]$$

Figure 2.8: F-structure for “Who did Mary see?” before (a) and after long distance dependency resolution on the f-structure (b)

The resolved f-structure in Figure 2.8 (b) has the reentrancy between the FOCUS and OBJ functions and also the SUBJ function of the main PRED and the XCOMP indicated by their shared values.

A full description of the annotation algorithm, its components and its operation on treebank trees and parser output trees can be found in McCarthy (2003), Cahill (2004) and Burke (2006). Below, I briefly describe the two parsing architectures presented in Cahill et al. (2002b) and Cahill et al. (2004).

Parsing Architectures Using the F-structure Annotation Algorithm

Cahill et al. (2002b, 2004) present two architectures for parsing raw text into f-structures using the automatic f-structure annotation algorithm: the pipeline and integrated models as shown in Figure 2.9.

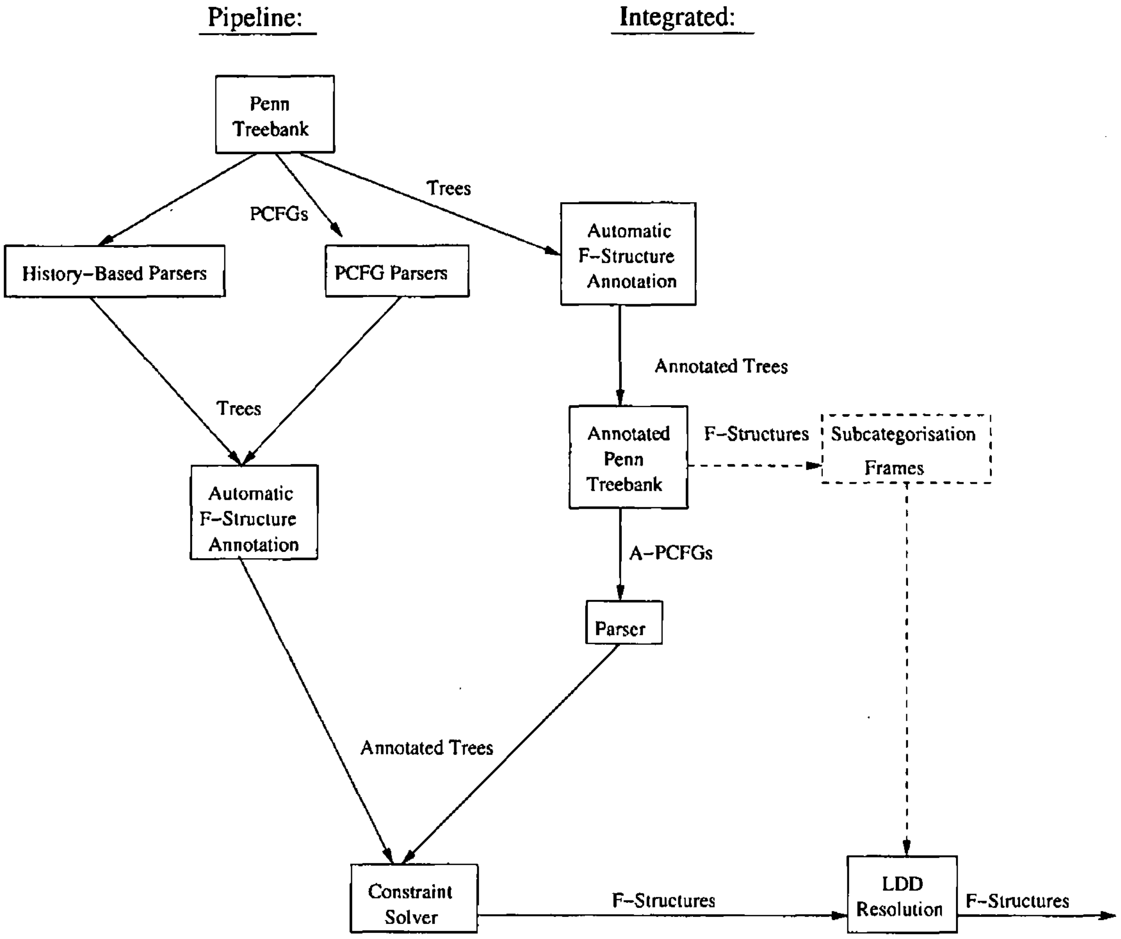


Figure 2.9: Two parsing architectures for parsing text into f-structures

In the pipeline model a PCFG or a history-based parser is extracted from (trained on) the Penn-II Treebank which is then used to parse raw text. The parser output is then passed to the automatic f-structure annotation algorithm which annotates the trees with functional equations. These are then passed to a constraint solver and the LDD-resolution module which produce f-structures.

In the integrated parsing model the training corpus is automatically f-structure annotated in a pre-processing step. After this pre-processing, each node in the c-structure trees has associated functional annotations like those in Figures 2.2 and 2.3 An annotated PCFG (APCFG) with rules of the form

$$
\begin{array}{l}
\mathrm{S}[\uparrow=\downarrow] \rightarrow \mathrm{NP}[\uparrow \textsc{subj} =\downarrow] \;\; \mathrm{VP}[\uparrow=\downarrow] \\
\mathrm{NP}[\uparrow \textsc{subj} =\downarrow] \rightarrow \mathrm{DT}[\uparrow \textsc{spec:det} =\downarrow] \;\; \mathrm{NN}[\uparrow=\downarrow] \\
\mathrm{VP}[\uparrow=\downarrow] \rightarrow \mathrm{V}[\uparrow=\downarrow] \;\; \mathrm{NP}[\uparrow \textsc{obj} =\downarrow] \\
\vdots
\end{array}
$$

is extracted from the functionally annotated training corpus. The parser parses raw text using the APCFG to produce f-structure annotated trees. The annotations are then sent to the constraint solver and LDD resolution module which produce f-structures.

Evaluation of LFG Parsing Architectures

The output generated by the LFG Parsing Architectures is evaluated in a number of ways. The c-structure trees are evaluated to assess the quality of the syntactic analysis. The f-structure dependencies are evaluated to assess the quality of the functional analysis. Dependency evaluations against gold standard references are usually conducted in two forms: all grammatical functions, and predicates only evaluations. All grammatical functions evaluations calculate precision, recall and f-score on all of the attribute:value pairs in the f-structures. Predicates only (preds-only) f-structure evaluations consider a subset of the grammatical functions in the f-structures by stripping out grammatical attributes, e.g. tense, case and number, that are not directly relevant to the basic predicate-argument structure of the f-structure. Preds-only evaluations ignore all paths through the f-structure that do not end in a PRED attribute:value pair. While this is a less comprehensive evaluation metric, it focuses on the core predicate-argument-adjunct structure and ignores "easier" attribute:value pairs, given a structurally flawed f-structure, which may result in an artificially high evaluation score.

Where a hand-crafted gold standard test set is lacking, or only a small hand-crafted gold standard exists, many researchers (Hockenmaier and Steedman, 2002; Miyao et al., 2003; Cahill, 2004; Judge et al., 2005) use automatically generated gold standards to test against. While these gold standards are not as good a resource as a hand-crafted gold standard, they have the advantage that they can be very easily created automatically. Following Hockenmaier and Steedman (2002), experiments with these kind of evaluations are referred to as CCG-style evaluations, where the original Penn-II Treebank Section 23 trees are automatically converted into Combinatory Categorial Grammar (CCG) derivations, which are then used to evaluate CCG parser output for the same strings.

Comparison of Pipeline and Integrated LFG Parsing Models

Table 2.4 shows dependency-based evaluation results for testing these two parsing models against the DCU 105, and in a CCG-style experiment similar to that of Hockenmaier and Steedman (2002) against 2416 automatically generated f-structures for Section 23 of the Penn-II Treebank. The DCU 105 (Cahill et al., 2002b) dependency gold standard is a subset of 105 sentences taken randomly from Section 23 of the Penn-II Treebank, for which gold standard f-structures were generated by hand. In each case, the grammars are extracted/trained on Sections 02-21 of the Penn-II Treebank. The results show that both PCFG models perform similarly, with the integrated model scoring better on the preds-only evaluations for both test sets with f-scores of 74.80% and 75.33% respectively, and the pipeline model scoring better in the all grammatical functions evaluations with f-scores of 84.02% and 84.00% respectively. Table 2.4 also shows results for Collins', Charniak's and Bikel's parsers in the pipeline model (Figure 2.9). The results show that the more sophisticated parsing models yield higher results than the simpler PCFG models.

		DCU 105		WSJ 2416	
		Preds Only	All GFs	Preds Only	All GFs
Model/Parser		F-Score	F-Score	F-Score	F-Score
Integrated	PCFG	74.80	81.20	75.33	82.72
Pipeline	PCFG	74.00	84.02	73.78	84.00
Pipeline	Collins	77.86	85.66	80.10	86.82
Pipeline	Charniak	80.50	86.75	82.63	88.08
Pipeline	Bikel	79.73	86.80	82.35	88.23

Table 2.4: LFG parsing results against the DCU 105 for the 2 LFG parsing models using different parsers

2.4 Penn-II Treebank Annotation of Questions

The Penn-II Treebank (Marcus et al., 1993) is a part-of-speech (POS) tagged and parse-annotated treebank of sentences. The POS tagging and parse annotation of the text is indicated by nested labelled bracketing to show which POS tag is associated with each word and the structure of phrases, subsentential clauses and the sentence itself. Figure 2.10 shows the POS annotation for the sentence "John saw Mary."

```
(NNP John) (VBD saw) (NNP Mary) (.  .)
```

Figure 2.10: POS annotation of "John saw Mary."

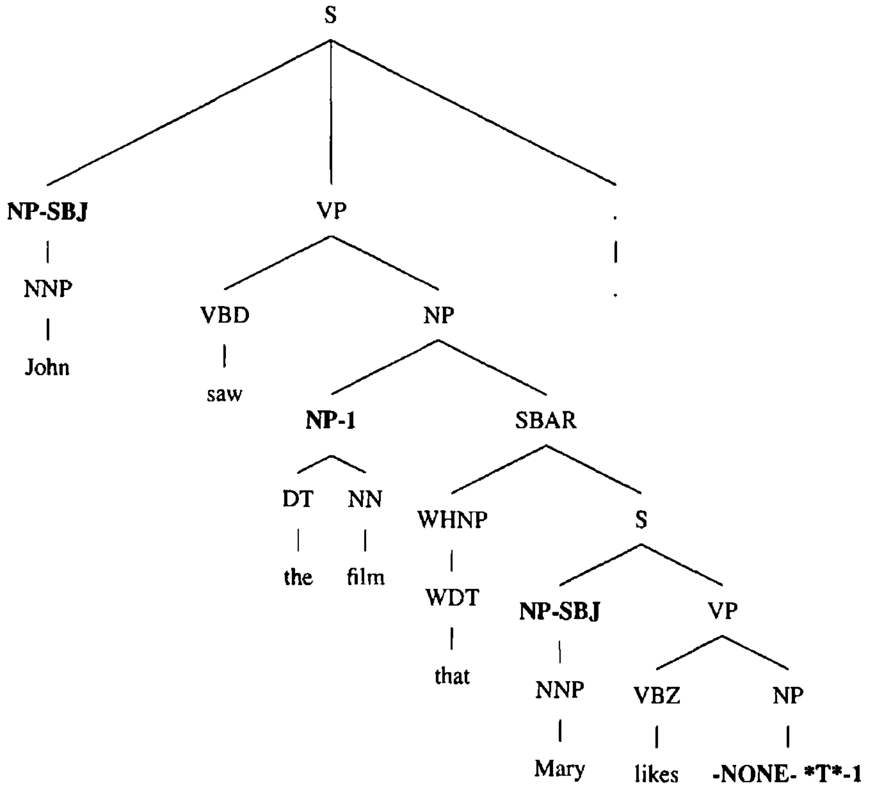
Note that both punctuation and lexical items are associated with a POS tag. Sentences are annotated with phrasal and clausal structure (Bies et al., 1995). Figure 2.11 shows the full POS-tag and parse-annotations for the sentence "John saw Mary."

Similar to the POS annotation, phrase level annotations like noun phrases (NP) and verb

```
(S (NP (NNP John)) (VP (VBD saw) (NP (NNP Mary))) (. .))
```

Figure 2.11: Parse annotation of "John saw Mary."

phrases (VP) as well as clause level annotations (S) are indicated through labelled bracketing of constituents or groups of constituents. In addition to labelling the constituents, Penn-II style annotation also includes some functional information like identifying the subject or topic. These are indicated by functional tags appended to the label of the governing node e.g. NP-SBJ. Movement phenomena and long distance dependencies (LDDs) are encoded by means of empty nodes in the tree (traces) where the dislocated element should be interpreted which are coindexed with their antecedent. Figure 2.12 shows the annotation of the sentence "John saw the film that Mary likes" with functional annotation and trace information and the corresponding tree structure.



```
(S (NP-SBJ (NNP John)) (VP (VBD saw) (NP-1 (NP (DT the) (NN film)) (SBAR
(WHNP (WDT that)) (S (NP-SBJ (NNP Mary)) (VP (VBZ likes) (NP (-NONE-
*T*-1))))))) (. .))
```

Figure 2.12: Functional tags and trace information in Penn-II Treebank annotation

Note that in Figure 2.12 there are two functional tags (-SBJ) identifying the subject of the main and embedded clause. The constituent "the film" is interpreted as the object of the verb "likes" in the embedded clause. This is indicated by the empty production -NONE- and the trace *T*-1 coindexed with the NP-1 antecedent ("the film") in the main clause.

The Penn-II Treebank annotation scheme (Bies et al., 1995) provides a number of special syntactic labels for dealing with question constructions. They are:

SBARQ Direct question introduced by a wh-word or a wh-phrase. Indirect questions and relative clauses should be bracketed as SBAR, not SBARQ.

SQ Inverted yes/no question, or main clause of a wh-question, following the wh-phrase in SBARQ.

WHADJP Wh-adjective Phrase. Adjectival phrase containing a wh-adverb, as in how hot.

WHAVP Wh-adverb Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing a wh-adverb such as how or why.

WHNP Wh-noun Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing some wh-word, e.g. who, which book, whose daughter, none of which, or how many leopards.

WHPP Wh-prepositional Phrase. Prepositional phrase containing a wh-noun phrase (such as of which or by whose authority) that either introduces a PP gap or is contained by a WHNP.

WDT Wh-determiner

WP Wh-pronoun

WP$ Possessive wh-pronoun (prolog version WP-S)

WRB Wh-adverb

Figure 2.13: Penn-II Treebank constituent labels for questions

I will give examples of some of these labels and their usage in bracketed question structures below. A full list of Penn-II Treebank bracket labels and functional tags can be found in

Appendix A.

The annotation of empty elements is particularly important when analysing questions into c-structure trees. In Penn-II style annotation wh-questions are generally analysed as involving an empty element which corresponds to the focus (the wh-element).[7] Figure 2.14 shows the trees for the question "Who saw the film?" and the corresponding statement "John saw the film." Note the similarity between the subtree rooted at SQ in the question tree and the tree for the statement.

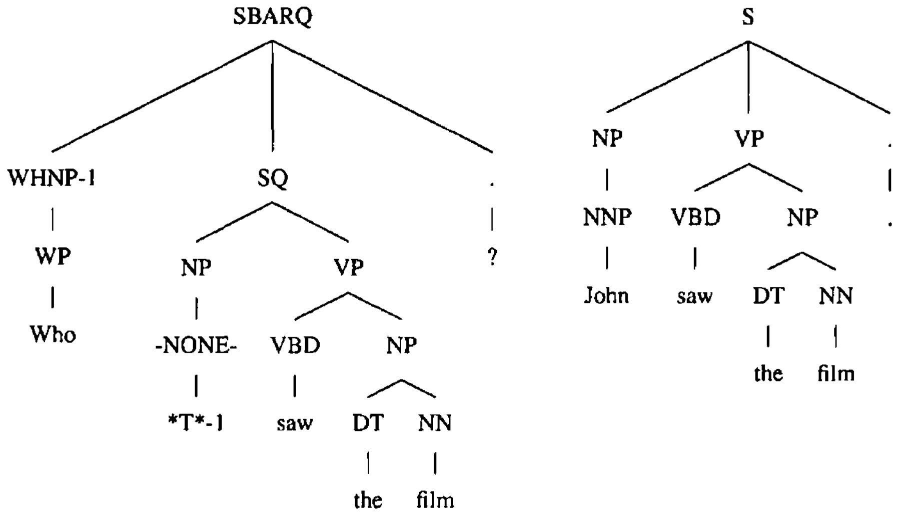


Figure 2.14: Parse annotated trees for a question and corresponding declarative statement

Note that while the structure of the tree for the declarative sentence is almost identical to that of the subtree rooted at SQ in the question, there is a crucial difference indicated between the two by labelling one as an S node and the other as SQ. The sentence expressing the statement consists of a simple declarative clause, i.e. one that is not introduced by a (possible empty) subordinating conjunction or a wh-word and that does not exhibit subject-verb inversion. By contrast, in the question, the corresponding SQ subtree is the main clause of a wh-question, following the wh-phrase in SBARQ.

[7] In some syntactic theories (Chomsky, 1973), interrogative sentences are derived from corresponding declarative sentences.

The original position of the dislocated element of the wh-movement in Figure 2.14 is adjacent to its antecedent in the tree. An example of a "real" long distance dependency in a question is shown in Figure 2.15.

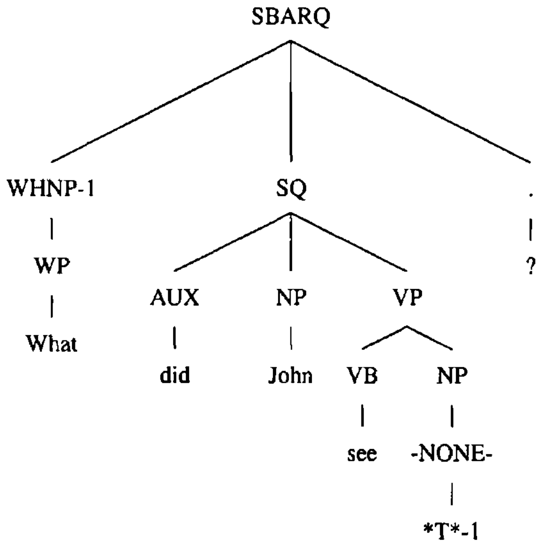


Figure 2.15: Long distance dependency in a question

An interesting (and quite) common question structure involving LDDs is one involving copular constructions with the verb "be." In contrast to other constructions where the NP following the verb is the object of the verb, in copular question constructions the NP following the copular verb (be) is the subject of the sentence. The overt NP inside the SQ in Figure 2.15 is in fact the subject of the sentence (despite being to the right of the main verb) and the dislocated element appears at the end of the clause. According to the Penn-II Treebank bracketing guidelines (Bies et al., 1995) there is no VP node inside an SQ in copular questions. Figure 2.16 shows the tree for the question "What is John?"

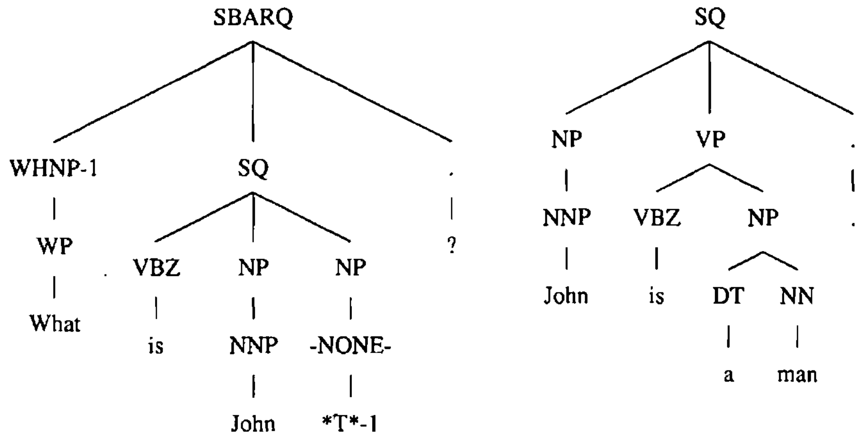


Figure 2.16: Penn-II analysis for a copular question and corresponding declarative sentence

2.5 NLP in QA

The level of usage of NLP techniques in QA varies from system to system with some more dependent on NLP than others. Also the depth of NLP techniques used varies, with some systems using only shallow lexical processing or chunking with others opting to use full syntactic parsing or to derive deeper dependency structures.

Harabagiu et al. (2000a,b) describe an NLP-rich QA system which combines both shallow and deep processing. In this system questions and answers are subject to syntactic and semantic analysis, while the documents retrieved for answer extraction receive only shallow linguistic processing. Using this method, their system achieves a very high score on the TREC-8 test set, achieving best results (89.5% precision and 84.75% NIST) when run in its most NLP intensive configuration which performs semantic transformations on both question and answers for an inference-based answer justification routine.

At the core of the NLP-rich question and answer processing of Harabagiu et al. (2000a,b) is a CFG parser (Collins, 1996). The parser output is used to generate logical form semantic representations, to determine the question class (what type of named entity it queries for), and also for expanding and reformulating the question to increase the coverage of the document retrieval process.

Similarly Katz et al. (2005) describe a number of linguistically motivated techniques for question answering using syntax, semantics, and predicate logic representations. Their architecture uses syntactic analysis to decompose complex or ambiguous segments of questions. For example the question "When was the 20^{th} President of the U.S. born?" is decomposed into identifying the 20^{th} President of the U.S. and then when he was born. This allows their system to narrow the search space by only considering documents relevant to the president in question as opposed to all presidents.

NLP methods like CFG parsing can also be used in answer selection and ranking. Tree distance, which assigns an associated cost to transforming one CFG tree to another, is one metric which can be used to select and rank answer candidates which have a tree distance less than a threshold value when compared with the question. This is a method which has been used for answer retrieval and proposed as a means to evaluate parsers by Emms (2005a,b). There is also scope for taking this method further using deeper linguistic representations such as f-structure (or similar dependency graphs or logical forms). F-structures are graphs and a graph edit distance metric can be used to compare similarity between answer candidates and questions.

2.6 Summary

In this chapter I have introduced Question Answering as a distinct but related field to Information Retrieval. QA has been researched since at least the early 1960s, with early successes as database frontend and closed-domain text-based systems. As a research area, QA is growing, expanding into cross language and multilingual domains and also into retrieving information from different media types (MMQA).

I gave a brief overview of context-free grammars, probabilistic context-free grammars and parsing and three state-of-the-art history-based, lexicalised, generative probabilistic parsers (Collins, 1999; Charniak, 2000; Bikel, 2002). I introduced Lexical Functional Grammar (Kaplan and

Bresnan, 1982), outlined recent work on automatic f-structure annotation (Cahill et al., 2002a, 2004) and automatic f-structure annotation-based parsing models (Cahill et al., 2002b).

I introduced the Penn-II Treebank data-structures and encoding of linguistic information relevant to question material.

I briefly discussed how deep NLP methods are used in state-of-the-art QA systems. I also outlined current research in answer detection and ranking using deeper linguistic analysis of question and answer strings.

Chapter 3

Domain Variance Experiments with the ATIS Corpus

3.1 Introduction

This chapter presents work on using syntactic parsers in the domain of question analysis. I show that the ATIS corpus data (Hemphill et al., 1990) is substantially different from the financial newspaper-style text found in the Wall Street Journal sections of the Penn-II Treebank, and that the relatively high proportion of questions in ATIS makes it suitable for initial tests of parser performance in the question domain. Statistical treebank-based parsing resources reflect the properties of their training data and generally underperform on data significantly different from that upon which they were trained. I present baseline experiments to show that this is the case for three state-of-the-art history-based parsers trained on Penn-II Treebank data and tested on the ATIS corpus. I show how retraining the parsers on a training set which includes some ATIS material boosts the performance significantly on the "out-of-domain" ATIS data.

Section 3.2 presents previous work on statistical parsing and domain variance. In Section 3.3, I compare and contrast the ATIS corpus with the Penn-II Treebank and give a selection of typical example sentences from each. I describe my baseline parsing experiments and results in Section 3.4. Section 3.5 describes experiments with retraining the parsers on a training corpus which includes ATIS data. These experiments produce significantly better results than the baseline in tests on a test set taken from the ATIS corpus. Section 3.6 summarises and concludes.

There are a number of options for examining parser performance given the training and evaluation corpora available (Penn-II Treebank and ATIS). The space of possibilities explored in the experiments in this chapter is indicated below, with asterisks (*) indicating areas of the experimental space examined in this chapter.

Test Training	Penn Sect 23	ATIS (all)	ATIS 10%
Penn	*	*	*
ATIS			*
ATIS + Penn			*

In the training column, Penn refers to training on Sections 2-21 of the Penn-II Treebank, ATIS to training on 90% of the ATIS corpus (only) and ATIS + Penn to training on 90% of ATIS and Penn-II Treebank data. The tests are Section 23 of the Penn-II Treebank, the entire ATIS corpus, and a 10% sample of ATIS withheld as a test set.

Earlier versions of the results presented here have been published in Judge et al. (2005). To the best of my knowledge this is the first research to study the effect of domain variation that establishes the statistical significance of the results.

3.2 Previous Work

Gildea (2001) studied the effects of corpus variation on parser performance by testing a parser trained on the Penn-II Treebank on the Brown corpus. He observed that the parser performance dropped by 5.7% labelled bracketing f-score when the domain was varied in this way. This shows that the effects of domain variance are evident even if the out-of-domain corpus is not *drastically* different from the original training corpus (both the Penn-II and Brown corpora consist primarily of written texts of American English, the main difference is the more varied nature of the text in the Brown corpus). The performance drop due to domain variance was remedied by adding appropriate data to the parser's training corpus, but Gildea notes that a large amount of additional training data has little effect on the results if it is not matched to the test data.

In the question domain, Clark et al. (2004) have worked with parsing questions using Combinatory Categorial Grammars (CCGs). Their experiments focused on "What ...?" questions from the TREC QA testsets. They retrain the CCG lexical supertagger to cope with the new domain instead of retraining their whole parser. This is because previous work (Clark and Curran, 2004) has shown that a high lexical tagging accuracy is sufficient to produce good CCG parsing results. Their work improves the supertagger accuracy on their "What" question corpus by over 13%.

3.3 Corpus Description and Comparison

This section briefly describes and compares the Penn-II Treebank (Marcus et al., 1993) and the ATIS corpus (Hemphill et al., 1990).

3.3.1 The Penn-II Treebank

The Wall Street Journal (WSJ) sections of the Penn-II Treebank (Marcus et al., 1993) consist of approximately 1 million words (50,000 sentences) of American English taken from Wall Street Journal articles in 1989. The sentences are POS tagged and parse-annotated according to guidelines set out in Santorini (1990) and Bies et al. (1995). An important distinction of the Penn-II Treebank from the first release (Penn-I Treebank Marcus et al. (1993)) is that the trees have been annotated with some non-context free information in the form of functional tags (-SBJ (subject), -LOC (locative), etc.) which indicate semantic roles, and empty nodes and coindexation to indicate long distance dependencies (Marcus et al., 1994).

3.3.2 ATIS

The Air Travel Information System (ATIS) corpus (Hemphill et al., 1990) is a transcription of spoken dialog with an automated air travel information system. The ATIS corpus used in the research presented here is that distributed with the Penn Treebank release 2 (Penn-II Treebank). The ATIS corpus consists of 578 sentences which are POS tagged and annotated according to Penn-II Treebank guidelines. ATIS data represents a different style of language from the Wall Street Journal texts of the Penn-II Treebank: a significant proportion of the sentences in ATIS are questions, there are imperatives and non-sentential utterances, which are generally shorter than those in the WSJ sections of the Penn-II Treebank and the transcription does not contain punctuation marks.

1. `Are there any flights arriving after eleven a.m`

2. `Show me the T W A flight`

3. `I need a flight from Los Angeles to Charlotte today`

4. `Flights from Los Angeles to Pittsburgh`

5. `On Tuesday arriving before five p.m`

6. `What flights from Philadelphia to Atlanta`

Figure 3.1: Example ATIS utterances

Figure 3.1 illustrates typical ATIS corpus data including both question (1) and non-question sentences (2,3), as well as sub-sentential (4,5) and incomplete utterances (6). Note also that punctuation is not included in the ATIS strings.

3.3.3 Penn-II WSJ vs. ATIS

Both Penn-II WSJ and ATIS are POS- and parse-annotated following the same general annotation guidelines (Bies et al., 1995). Despite these similarities, the two treebanks exhibit strong differences as regards size, domain, phrase type distribution and punctuation.

	ATIS	Penn-II WSJ
Words	4000 words	1,000,000 words
Sentences	578 sentences	50,000 sentences
Average sentence length	7 words	21 words
Source	Transcription of spoken dialog	WSJ Newspaper text
#Questions	213 Direct questions	233 Direct questions
Sentence type	Interrogatives, imperatives, and fragments	Declarative sentences
Inter-Word Punctuation	None	Punctuated

Table 3.1: Corpus statistics compared

Table 3.1 shows a comparison of the Penn-II WSJ sections and the ATIS corpus. The most striking difference between the Penn-II Treebank WSJ sections and the ATIS is the difference in size between the two corpora: the WSJ sections of the Penn-II Treebank with 50,000 sentences are over eighty times the size of ATIS with only 578 sentences. Another important difference between the two is the average sentence length: the sentences in ATIS tend to be much shorter than the WSJ Sections of the Penn-II Treebank, with an average length of 7 words, compared to 21 words in the WSJ Sections. Figure 3.2 plots the number of sentences against the sentence length for the ATIS corpus and Section 23 of the WSJ section of the Penn-II Treebank illustrating the difference in sentence length distribution between the two corpora.

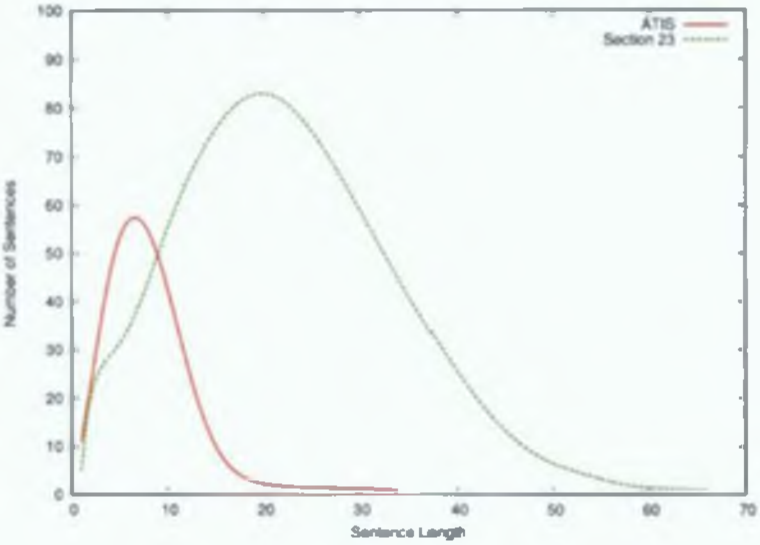


Figure 3.2: Sentence length distributions ATIS vs WSJ Section 23 (Bezier interpolated)

The graph shows how significantly larger a single section of the Penn-II Treebank WSJ sections is than ATIS. It also shows the broader distribution of data over the sentence lengths in the section of the Penn-II Treebank, which has a much wider spread over the sentence lengths. Section 23 has a mean sentence length of 21 words with a standard deviation of 8.6, while ATIS has a mean sentence length of 7 words with a standard deviation of 2.9.

The source of the text for the two corpora also highlights some important differences. The source for the ATIS corpus is spoken dialogue which tends to be more casual and brief (Figure 3.1) than the longer, more complex sentential structures found in the Penn-II Treebank (Figure 3.3). Also, the nature of the air travel information system results in the ATIS corpus containing sentences of a largely information seeking and interrogative nature. Of the 578 sentences in the ATIS corpus, 213 are questions or interrogatives, accounting for over 36% of the entire corpus. Comparatively, the Penn-II WSJ sections have very few interrogative sentences or questions, only 233 over the entire WSJ sections (accounting for less than 0.5% of the entire corpus). In addition, many of the Penn-II questions are embedded or rhetorical questions (Figure 3.4 (3)), which unlike those in the ATIS do not seek information. Interestingly, none of the 233 direct questions in the WSJ sections are to be found in Section 23 of the treebank, which is the standard testing section for parser evaluation. Therefore, none of the previous evaluations in the literature carried out on this section reflect the quality of parsing question data.

1. Shares of UAL, the parent of United Airlines, were extremely active all day Friday, reacting to news and rumors about the proposed $6.79 billion buy-out of the airline by an employee-management group.

2. Ports of Call Inc. reached agreements to sell its remaining seven aircraft to buyers that weren't disclosed.

3. As a group, stock funds held 10.2% of assets in cash as of August, the latest figures available from the Investment Company Institute.

Figure 3.3: Example Penn-II Treebank WSJ sentences

1. For example, what exactly did the CIA tell Major Giroldi and his fellow coup plotters about U.S. laws and executive orders on assassinations?
2. Who'd have thought that the next group of tough guys carrying around reputations like this would be school superintendents?
3. What is the way forward?
4. But if rational science and economics have nothing to do with the new environment initiative, what is going on?

Figure 3.4: Example Penn-II Treebank WSJ questions

3.4 Baseline Experiments

In my baseline experiments I use three state-of-the-art Penn-II Treebank trained parsers (Collins, 1999; Charniak, 2000; Bikel, 2002) to parse the ATIS corpus. The parsers are not modified or retrained in any way for this task, in order to determine how well these parsers can or cannot cope with data from the question-rich ATIS corpus.

3.4.1 Evaluation Tools and Metrics

I evaluate the parsing experiments in this section using standard PARSEVAL precision and recall metrics (Black et al., 1991) and, where it is less than 100% of the test set, parser coverage, which is the percentage of the test set for which the parser successfully produces a complete spanning parse. Precision and Recall are calculated as follows: given P, the proposed analysis from the system and T, the gold standard analysis,

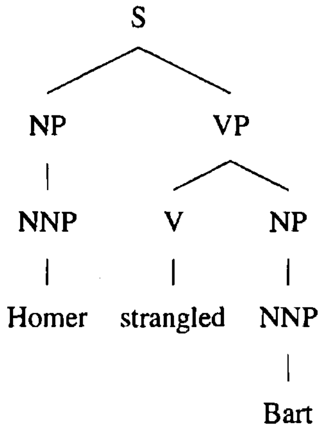
$$Precision = \frac{number\ of\ correct\ constituents\ in\ P}{number\ of\ constituents\ in\ P} \tag{3.1}$$

$$Recall = \frac{number\ of\ correct\ constituents\ in\ P}{number\ of\ constituents\ in\ T} \tag{3.2}$$

The f-score (harmonic mean) of precision and recall is calculated by

$$F - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.3}$$

I use the `evalb`[1] scoring program designed by Sekine and Collins to calculate these scores. `evalb` takes a bracketed representation of gold standard and test trees (Figure 3.5) and outputs precision, recall, f-score, bracket crossing, and tagging accuracy information (Figure 3.6). Unless otherwise stated, all results reported in this thesis are labelled precision, recall and f-scores.



```
(S (NP (NNP Homer))
(VP (V strangled) (NP (NNP Bart))))
```

Figure 3.5: Parse tree and corresponding `evalb` bracketing representation

[1] Available at http://nlp.cs.nyu.edu/evalb/

```
  Sent.                        Matched  Bracket   Cross        Correct Tag
 ID  Len.  Stat. Recal  Prec.  Bracket gold test Bracket Words  Tags Accracy
============================================================================
   1    6    0   55.56  71.43     5      9    7     1      6      6  100.00
   2    6    0   87.50 100.00     7      8    7     0      6      6  100.00
   3    4    0   80.00 100.00     4      5    4     0      4      4  100.00
   4    5    0   75.00  85.71     6      8    7     0      5      5  100.00
   5    6    0   83.33 100.00     5      6    5     0      6      4   66.67
   6    8    0   36.36  40.00     4     11   10     1      8      6   75.00
   7    6    0   44.44  57.14     4      9    7     1      6      4   66.67
   8    4    0   50.00  60.00     3      6    5     1      4      4  100.00
   9    7    0   75.00 100.00     6      8    6     0      7      6   85.71

                                       :

============================================================================
                 61.11  69.06    308    504   446     35     441   401     90.93
=== Summary ===

-- All --
Number of sentence        =     58
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =     58
Bracketing Recall         =  61.11
Bracketing Precision      =  69.06
F-Score                   =  64.84
Complete match            =   0.00
Average crossing          =   0.72
No crossing               =  60.34
2 or less crossing        =  94.83
Tagging accuracy          =  90.93

-- len<=40 --
Number of sentence        =     58
Number of Error sentence  =      0
Number of Skip  sentence  =      0
Number of Valid sentence  =     58
Bracketing Recall         =  61.11
Bracketing Precision      =  69.06
F-Score                   =  64.84
Complete match            =   0.00
Average crossing          =   0.72
No crossing               =  60.34
2 or less crossing        =  94.83
Tagging accuracy          =  90.93
```

Figure 3.6: Sample `evalb` output

Statistical Significance Testing

When comparing experimental results it is useful to be able to say if differences between two sets of results are statistically significant, that is to say that the difference has not occurred by chance. I use approximate randomisation to test for statistical significance between sets of results. Approximate randomisation is a computationally-intensive randomisation test (Noreen, 1989) which can be applied to non-linear functions of variables such as precision and f-score.

An approximate randomisation test tests the null hypothesis that the difference between two data samples occurred by chance. This is done by calculating the difference between their mean values: if the null hypothesis holds, then randomly shuffling values between the two samples should make little difference between their mean values. An exact randomisation test does this for all possible permutations, an approximate randomisation test (like the one used here) estimates this by performing n randomisations on the data samples. The accuracy of the approximate randomisation test depends on the value of n. The test calculates a p-value, which is the probability that the null hypothesis is true. A low p-value indicates that the null hypothesis is likely to be false, therefore the difference between the two results is likely not to be by chance and therefore statistically significant.

I perform approximate randomisation testing using a perl script by Bikel[2] which uses the output of `evalb` to evaluate the significance of new results for parse trees. In Chapter 4 I perform similar testing on f-structures. The software to perform this test was provided by Stefan Riezler (through personal communication) and performs the approximate randomisation testing on sets of triple encoded f-structures (Crouch et al., 2002).

Unless otherwise stated, for the experiments in this thesis I take $n = 10000$ (the default value for Bikel's script) and a p-value of less than 0.05 as indication that the null hypothesis is false and an observed difference in scores is significant at the 95% level.

[2] Available at http://www.cis.upenn.edu/~dbikel/software.html

3.4.2 Experiments

In order to establish a baseline for parsing ATIS question data, I parsed all 578 ATIS sentences with 3 Penn-II trained, off-the-shelf, state-of-the-art history-based parsers (Collins, 1999; Charniak, 2000; Bikel, 2002) using the resources supplied in the distributions (trained on Penn-II Treebank Sections 02-21) and evaluate against the original ATIS treebank trees. The baseline results are shown in Table 3.2 along with the published results for the parsers tested on Section 23 of the Penn-II Treebank.

	ATIS			Section 23		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Collins Model 2	76.43	68.87	72.45	88.3	88.1	88.2
Charniak	62.19	65.16	63.64	89.5	89.6	89.55
Bikel's Emulation of Collins M2	74.79	65.68	69.94	88.2	88.3	88.25

Table 3.2: Baseline Parsing Results on the ATIS corpus

The results show that each of the parsers tested performs considerably worse on the ATIS corpus than in tests on Section 23 of the Penn-II Treebank. The average difference between the f-scores on the two test sets is 19.99% with Charniak's parser suffering the worst drop at 25.91% and Collins' suffering the smallest (though still considerable) drop at 15.75%.

The observed drops are considerably worse than those observed in previous domain variation experiments by Gildea (2001). The results confirm that the ATIS experiments presented here constitute a much stronger instance of domain variation than the Penn-II/Brown corpus experiments reported by Gildea (2001).

3.4.3 Punctuation

The Penn-II Treebank Wall Street Journal sections used for training the parsers contain properly punctuated text. On the other hand, the ATIS strings are unpunctuated. This is a factor that could possibly explain the underperformance of the parsers in the ATIS experiments, as we would expect grammars trained on Penn-II Treebank sections to perform better on punctuated text.[3]

To test the influence of punctuation on the ATIS parsing results, I manually added basic punctuation to each of the ATIS sentences (parser input and gold standard trees). Each of the 213 questions had a question mark added, the remaining sentences had a fullstop added, and the sub-sentential fragments were left unpunctuated. I then reran the baseline parsing experiments with each of the parsers. Table 3.3 shows the results for testing the parsers on a minimally punctuated version of ATIS along with the p-values for significance testing for each individual parser comparing its performance on the punctuated ATIS with that of the unpunctuated ATIS.

	ATIS			ATIS Punctuated			
	Precision	Recall	F-Score	Precision	Recall	F-Score	p-Value
Collins	76.43	68.87	72.45	76.35	68.87	72.42	0.3413
Charniak	62.19	65.16	63.64	66.71	69.60	68.12	9.99×10^{-5}
Bikel's Emulation of Collins M2	74.79	65.68	69.94	74.61	65.58	69.80	0.0025

Table 3.3: Parsing results for punctuated ATIS sentences

The results show a slight drop in performance for Collins' and Bikel's parsers, and an improvement of 4.48% f-score for Charniak's parser when tested on punctuated ATIS sentences. The increase in performance for Charniak's parser is statistically significant (p-value $\leq$ 0.05). The drop in performance for Bikel's parser is also statistically significant, however the smaller 0.03% drop in f-score for Collins' parser is not. It is interesting, though not entirely unexpected,

[3] This was pointed out by Tracy King (personal communication).

that Charniak's parser gains most from having punctuation added, as his parser treats punctuation less trivially than Collins' or Bikel's. The parsing model of Collins and Bikel treats punctuation differently to other surface elements of the sentence, and tries to attach it as high in the tree as possible so that it is positioned between two non-terminals.

It is evident that a lack of punctuation is responsible for some of the drop in performance on parsing ATIS data, but that the choice of parser can significantly influence the degree to which parsing is effected. Nevertheless, punctuation (or lack thereof) is not responsible for the performance drop being as severe as shown in Table 3.2. This suggests that the parsers are having difficulty coping with the ATIS question material given the training information extracted from the Penn-II Treebank.

3.5 Retraining Experiments

With the poor performance of Penn-II Treebank-trained state-of-the-art history-based parsers on ATIS data, clearly a parser fine-tuned for use on questions is needed if parsing questions is to be incorporated successfully as a stage in a QA system. Given that the parser is dependent on the type of data in the training corpus, the logical way to accomplish this is to retrain the parser on a question corpus. By retraining the parsers on a treebank which is more representative of the text to be parsed (in this case questions) we expect the parser will yield better results on the question corpus.

A number of corpora were available for use. The Penn-II Treebank, though not a question corpus, is a very large corpus of parsed English sentences, giving broad language coverage. The ATIS corpus as described above, is a small corpus of Penn-II-style parsed English questions as well as imperatives and some fragment data. Also available are the TREC QA track test questions and the Cognitive Computation Group at the University of Illinois[4] (CCG[5]) question

[4] http://l2r.cs.uiuc.edu/ cogcomp

[5] Note that the acronym CCG here refers to Cognitive Computation Group, rather than Combinatory Categorial Grammar mentioned in Section 3.2

classifier training set text corpora. These corpora amount to 2753 and 5452 raw unannotated questions, respectively. The history-based parsers require parse-annotated corpora as training resources. This makes the TREC QA and CCG question classifier corpora unsuitable as training data for these experiments, but nonetheless useful for further work (Chapter 5). For the experiments reported below, I therefore decided to use the parse-annotated ATIS resources.

Since the retraining experiments involve including ATIS data in the parser's training corpus, a dedicated test set of sentences needed to be held out of the ATIS material used for training the parsers. I randomly extracted a sample of 10% of the ATIS treebank trees to be held out as the gold standard test set against which to evaluate the retrained parsers. The results in Table 3.4 show the results from parsing the 10% of ATIS testset with the standard Penn-II trained resources for each of the parsers, compared with the results for parsing the whole of ATIS with the same parser. The results indicate that the randomly selected 10% test set is slightly more difficult to parse than the complete ATIS data.

	ATIS (All)			ATIS 10% Test Set		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Collins Model 2	76.43	68.87	72.45	73.17	65.48	69.11
Charniak	62.19	65.16	63.64	60.53	63.89	62.16
Bikel's Emulation of Collins M2	74.79	65.68	69.94	69.06	61.11	64.84

Table 3.4: Comparison of baseline parsing on all of ATIS and the 10% ATIS test set

I carried out four retraining experiments with slightly different parser training data on each run, testing on the same set of questions (a 10% subset taken at random from the ATIS corpus) withheld from the training data. The four training sets were constructed as follows:

- ATIS corpus (90%)

- ATIS corpus (90%) and Sections 02-21 of the Penn-II Treebank

- ATIS corpus (90%) and questions extracted from the whole Penn-II treebank
- Direct questions extracted from ATIS (90%) and questions extracted from the whole Penn-II treebank

Collins' parser does not come with functionality to retrain on a new corpus therefore no results are provided for Collins' parser in these experiments. Bikel's parser, however, emulates Collins' Model 2 parser and achieves similar results on WSJ text from the Penn-II Treebank (Table B.1).Therefore, Bikel's results can be interpreted as indicative of how Collins' Model 2 parser would perform in the retraining experiments.

	Charniak			Bikel		
Trained On	Coverage	F-score	p-Value	Coverage	F-Score	p-Value
Penn	100	62.16	-	100	64.84	-
ATIS	100	81.59	9.99×10^{-5}	100	85.20	9.99×10^{-5}
ATIS + Penn	100	78.73	9.99×10^{-5}	100	85.65	9.99×10^{-5}
ATIS + PennQs	100	84.69	9.99×10^{-5}	100	85.20	9.99×10^{-5}
ATISQs + PennQs	98.28	70.91	0.0055	100	73.06	0.0144

Table 3.5: ATIS test set parsing results for Bikel's and Charniak's parsers retrained using Penn-II Treebank and ATIS data

Table 3.5 compares the results for the parsers in retraining experiments using various combinations of ATIS and Penn-II Treebank data, testing on a random sample of 10% of ATIS which was held out of the ATIS section of the training data. The 10% sample used for testing was constant for all runs. The p-values shown are calculated by comparing the retrained parser's results with that of the same parser in the baseline experiment in Table 3.4. The results show that in each of the retraining runs, both parsers have gained a statistically significant improvement over the baseline results. Bikel's parser scores highest across all of the experiments, gaining over 20% f-score in all but the fourth run where only the questions from each data source are included in the parser's training set. In the fourth run the result for Bikel's parser is improved by 8.22%. Charniak's parser improves to a similar degree as Bikel's, achieving the biggest single gain in all of the runs at 22.53%, but does not better Bikel's results in any of the experiments.

It is interesting to note the difference between the two parsers in the ATIS and Penn trained run when compared with the ATIS trained run. For the ATIS + Penn trained run Charniak's parser's result has decreased by 2.86% compared to training on ATIS only, whereas Bikel's has increased by 0.45%. Despite the slight drop in performance when compared with training only on ATIS, Charniak's score of 78.73% in this run is still a dramatic increase over the baseline result (training on Penn only) of 62.16%. The test set here is quite small, and without further research it is difficult to say with any certainty what could be causing the different behaviour of the two parsers. However, one might speculate that the finding indicates that Charniak's parser is more sensitive to the larger amount of Penn-II Treebank data than ATIS, and that the positive impact of ATIS data in the training set is being "diluted" somewhat by the amount of extra Penn-II Treebank data.

These results show that both parsers' performance on ATIS material can be significantly improved by training exclusively on ATIS material or in combination with Penn-II Treebank data. Comparing Bikel's parser with Charniak's parser, Table 3.6 shows that Bikel's parser statistically significantly outperforms Charniak's in each of the experiments conducted. This is possibly an indication that, compared to Charniak's, Bikel's parsing model is easier retrained to be adapted for use in domains other than the Penn-II Treebank. This view is supported by the somewhat brittle nature of retraining Charniak's parser on new data which I experienced, and which has also been noted by other researchers.[6] It is interesting to note that the best result in these experiments, for both parsers, is achieved when the parser is trained on a combination of ATIS and some amount of Penn-II Treebank data. Also noteworthy is that the lowest result for both parsers is on the run where the least amount of ATIS material is included in the parser's

[6] Conversations with Aoife Cahill, Joachim Wagner and others at the NCLT revealed that we have each, independently, experienced similar difficulties when retraining Charniak's parser.

training set.

	F-score Difference	p-Value
ATIS	3.61	0.0002
ATIS + Penn	6.92	0.0002
ATIS + PennQs	0.51	0.0033
ATISQs +PennQs	2.15	0.0002

Table 3.6: Difference between Charniak's and Bikel's parsers' f-scores in retraining experiments and statistical significance testing

The improvement over the baseline reported in Table 3.5 is quite significant. As noted in Section 3.4.3, a small part of the reason for the low baseline score in the first place is the lack of punctuation in ATIS. Domain variance has been claimed to be responsible for the rest of the drop in performance. However, two important further characteristics of the ATIS corpus could also be affecting the scores and have yet to be discounted. Being a question-rich corpus, the ATIS data contains sentences of a much shorter length than those in the Penn-II Treebank. Also since ATIS is a transcription of spoken language, the sentences contain discontinuities and ungrammatical data, and these appear as constituents labelled FRAG and X in the parse-annotated corpus. Both of these factors, sentence length, and FRAG/X constituents, need to be considered as possible causes for the low baseline score before the results in Table 3.5 can properly support the conclusion that the low baseline is mainly caused by the domain variance in the question-rich corpus.

Figure 3.7 shows an analysis of Bikel's parser on both WSJ Section 23 of the Penn-II treebank and ATIS which plots f-score against sentence length. Figure 3.8 shows the same analysis for Charniak's parser. In both graphs the regions of statistical significance for each corpus are marked with vertical lines from the x-axis. These correspond to 2 standard deviations from the mean sentence length for each corpus, which, assuming roughly normal distribution of the data, should correspond to approximately 95 % of the data. For the ATIS corpus this is the region between sentence lengths 2 and 13 words, and for WSJ Section 23 this corresponds to the region between sentence lengths 5 and 40 words.

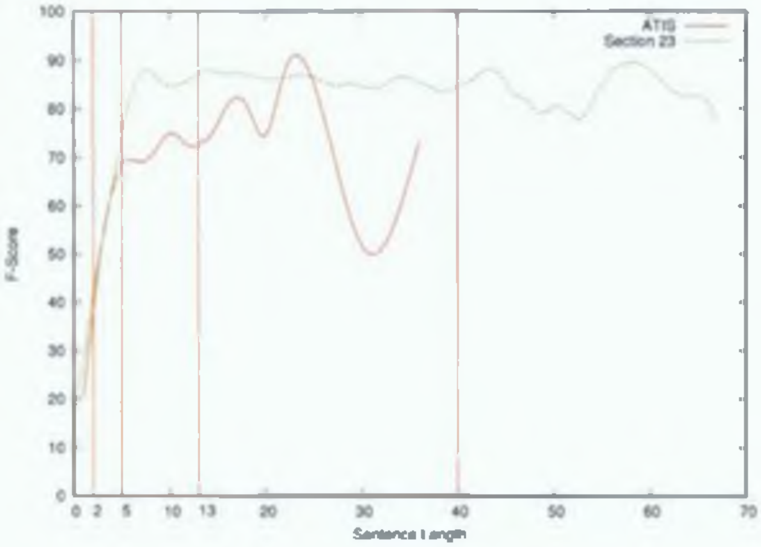


Figure 3.7: F-Score by sentence length comparison for Bikel's parser on WSJ Section 23 and ATIS

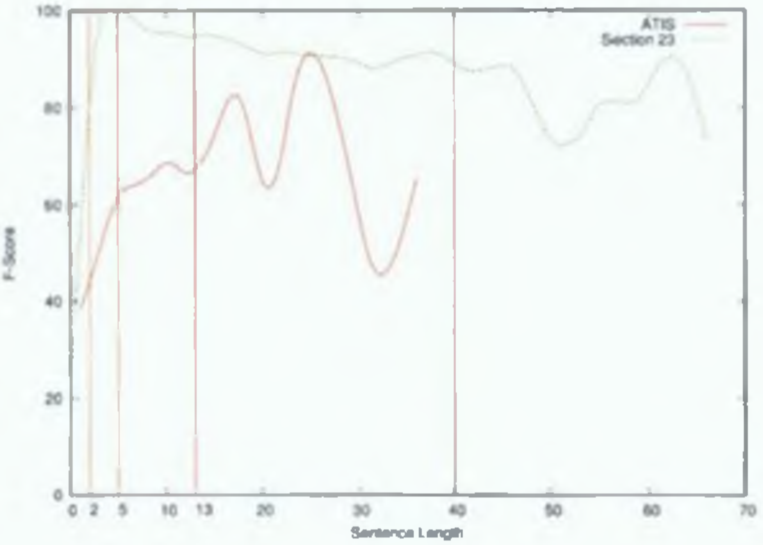


Figure 3.8: F-Score by sentence length comparison for Charniak's parser on WSJ Section 23 and ATIS

The area of overlap for the areas of significance for both corpora is the region between sentence lengths 5 and 13 words. There is a large gap between the two plots for both parsers. In both Figure 3.7 and 3.8 both parsers can clearly perform well on sentence lengths within this window, as indicated by the plot of WSJ Section 23 results. However the plot for ATIS results within this window shows considerably lower scores. This shows that the parsers can perform well on short sentences, but that they have difficulty with short sentences from a new domain, in this case ATIS.

In order to examine the effects of FRAG and X constituents on the results, I revisited the baseline results for Bikel's and Charniak's parsers and eliminated sentences which contain such constituents. In the Penn-II Treebank bracketing guidelines these are constituents which are incomplete (FRAG) or which are unknown, uncertain or unbracketable (X) and problematic for human annotators, so they are certainly likely to cause difficulties for the parsers. Table 3.7 shows the results for Bikel's and Charniak's parsers tested on the whole ATIS corpus, ignoring sentences containing FRAG/X constituents and comparing with the original ATIS baseline results in Table 3.2.

	ATIS (All)			ATIS (No FRAG/X)		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Bikel	74.79	65.68	69.94	76.24	67.86	71.81
Charniak	62.19	65.16	63.64	68.35	65.88	67.09

Table 3.7: Comparison of parsing results for Bikel's and Charniak's parsers on the ATIS corpus excluding sentences containing FRAG/X constituents

The results in Table 3.7 shows that removing the sentences which contain FRAG/X constituents from the evaluations increases the results by a small amount, 3.45% for Charniak's parser, 1.87% for Bikel's. This shows that that the presence of FRAG and X constituents in the data is having a negative effect on the baseline results and so is contributing to the low baseline score. However, the increase in the results when these constituents are removed is small compared to the drop in performance caused by testing on ATIS data. Again, this shows that the low baseline results are mainly caused by domain variance. In Section 5.4.3 I provide examples of frequent analysis errors made by Bikel's parser on question data, which have a large effect on the evaluation results.

3.6 Conclusion

This chapter has shown that the Penn-II treebank, due to its very low question content, is unsuitable as an exclusive training resource for modern probabilistic parsers if they are to be used in analysis for QA. The ATIS corpus (Hemphill et al., 1990), while not a dedicated question corpus, contains a significant proportion of questions and constitutes quite a different domain to the Penn-II Treebank. The domain variance exhibited by the Penn-II/ATIS experiments is much stronger than that observed by Gildea (2001) in his Penn-II/Brown experiments, as is shown by the large drop in parser performance in the baseline experiments (Section 3.4). My experiments show that some of this drop in performance can be attributed to the absence of punctuation in the ATIS data, but this is only a small portion of the drop in performance, and only holds for one of the parsers tested. Similar to Gildea (2001), the results in Table 3.5 show that parsing results are not greatly improved when a large amount of data (Sections 02-21 of the Penn-II Treebank) not matched to the test set is added to the training set. Charniak's parser performs worse on the test set when trained on ATIS plus Sections 2-21 of the Penn-II Treebank than when trained only on ATIS data. Bikel's parser on the other hand performs slightly better, but the f-score gain is quite small (0.45%) when compared to the gain from retraining on ATIS data.

The retraining experiments show that, despite its relatively small size, sections of the ATIS corpus can be used effectively to retrain state-of-the-art parsers to cope with ATIS data. Quite large, statistically significant gains of over 20% labelled precision and recall f-score can be achieved by adding appropriate (here ATIS) training material to the parser's training corpus in order to allow it to cope with the new domain (ATIS), though a similar (and for Charniak's parser, greater) gain can be achieved by training on ATIS data alone. This is quite surprising given the size difference between the two corpora used (the Penn-II Treebank is over eighty times the size of the ATIS corpus) but also encouraging. The baseline result, though poor, shows that some of the information relevant to analysing questions correctly can be found in informative text like the WSJ. The significant gains in performance achieved through adding only a small amount of ATIS data indicate that a question corpus of similar size to the Penn-II Treebank is probably not needed in order to retrain a high-accuracy parser to perform optimally on question data.

The parsing experiments presented here use a rather small test set from the ATIS corpus. I have shown that while the ATIS corpus contains a relatively high proportion of questions, it is not a true question corpus. For example it does not contain any "Who...?" questions and contains a lot of fragment data. In order to properly evaluate performance on a wide variety of question types, a larger more representative test set is necessary. Chapters 5 and 6 present work on providing this.

The experiments carried out so far have only looked at the effect of porting Penn-II Treebank-based syntactic CFG parsing resources to the question domain. Given the experiments presented here and previous work in the area (Gildea, 2001; Clark et al., 2004) the question whether the same effects would be observed with deeper linguistic analysis such as LFG f-structures from Penn-II Treebank-based resources remains. Chapter 4 examines this.

Chapter 4

Domain Variance and Treebank-Based LFG Resources

4.1 Introduction

This chapter builds on the domain variance and retraining work presented in Chapter 3. I investigate the effect domain variance has on deeper analysis in the form of LFG f-structures produced automatically by the Penn-II Treebank-based f-structure annotation algorithm of Cahill et al. (2004). The expectation is that a drop in performance for c-structure (CFG) parsers using grammatical resources induced from the Penn-II Treebank, will cause a similar performance drop for the automatic f-structure annotation algorithm taking CFG trees produced by the parsers as input.

I show that the negative effects of domain variance on c-structure parsing observed in Chapter 3 are also observed in automatic f-structure annotation. However, the drop in performance is less pronounced for f-structure annotation than for c-structure parsing and can be remedied by simply retraining the c-structure parser as before. One perhaps surprising result is that the automatic f-structure annotation algorithm of Cahill et al. (2004) does not need any modification.

Section 4.2 provides some relevant background information. I describe the gold standards against which I evaluate, and my baseline experiments in Section 4.3. Section 4.4 describes retraining experiments to improve f-structure analysis. In Section 4.5 I present experiments to investigate further research questions about upper bounds, back-testing and parameterisation raised by the retraining work. Section 4.6 summarises and concludes.

As in Chapter 3 there are a number of options regarding training and testing sets, this time for both c-structure *and* f-structure evaluations. The table below shows what is covered in the experiments in this chapter.

Test Training	DCU 105	ATIS 100	ATIS (all)	ATIS Cross Validation
Penn	*	*	*	
ATIS				*
ATIS + Penn		*		
ATIS Cross Validation	*			*

Here Penn refers to training on all of Sections 2-21 of the Penn-II Treebank, ATIS to training on all of the ATIS corpus, ATIS + Penn to training on Sections 2-21 of the Penn-II Treebank and the ATIS corpus less 100 sentences which were held out as a test set (ATIS 100) and ATIS Cross Validation refers to training and testing on the whole ATIS corpus with 10-fold cross validation. The tests sets used are: the DCU 105, a gold standard test set of 105 sentences from PTB Section 23 which have been manually f-structure annotated and ATIS 100, a 100 ATIS sentence gold standard which has been manually f-structure annotated.

Some of the results published in this chapter have been published in Judge et al. (2005)

4.2 Background

The parser domain variance research of Gildea (2001) and the question retraining work of Clark et al. (2004) has shown that porting linguistic resources to a new domain is possible for both probabilistic CFG- and CCG-based parsers. In the previous chapter I have shown that parsing the ATIS corpus with state-of-the-art Penn-II Treebank-based probabilistic parsers represents an instance of stronger domain variance than that observed by Gildea (2001).

Burke et al. (2004), Cahill et al. (2004), and O'Donovan et al. (2004) present research on automatically producing LFG resources from treebanks. However, to date no research has been carried out to test the effect of domain variance on the treebank-induced LFG parsing resources of Cahill et al. (2004). Given that these resources are induced from the Penn-II Treebank and that the pipeline parsing architecture of Cahill et al. (2004) uses Penn-II Treebank trained c-structure parsers, the expectation is that performance will suffer in a similar way as was observed for the history-based CFG parsers in the previous chapter.

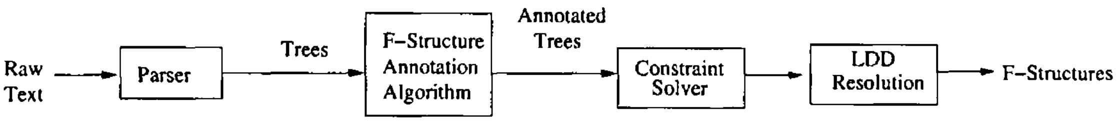


Figure 4.1: Pipeline Parsing Architecture of Cahill et al. (2004)

The automatic f-structure annotation algorithm of Cahill et al. (2004) described in Section 2.3.3 takes either parser output or treebank (c-structure) trees and annotates the trees with functional information based on a set of annotation rules originally designed for the Penn-II Treebank. The f-structure annotated trees are then passed to a constraint solver to produce f-structures (Figure 4.1). The pipeline parsing model entails that errors in early stages can be propagated to (and potentially be amplified by) modules further along the line. Because of this it is possible that the automatic f-structure annotation algorithm will suffer worse degradation of results than the c-structure parser because it is dependent on the parser output.

4.3 Baseline LFG Experiments

This section describes the baseline experiments to determine the portability of the resources of Cahill et al. (2004) to a new domain, the ATIS corpus. For these experiments I have chosen to use Bikel's parser because it is easily retrainable[1] and consistently outperforms Charniak's parser in the retraining experiments in Section 3.5.

4.3.1 F-Structure Gold Standard and Evaluation Tools

Before conducting experiments with the automatic f-structure annotation algorithm, a gold standard set of f-structures against which to evaluate the automatically generated f-structures is necessary. As in Chapter 3, I use a randomly extracted sample of the ATIS corpus. However, unlike the previous evaluations I carried out on ATIS data, additional annotation work needed to be done to generate an f-structure gold standard. 100 sentences were chosen at random from the ATIS corpus, and their corresponding treebank c-structure analysis was used as the gold standard against which to evaluate the c-structures in the experiments described in this chapter. The 100 gold standard c-structure trees were then passed to the automatic f-structure annotation algorithm to generate f-structures which were then manually corrected and compared by three human annotators.[2] These gold standard f-structures were then used in the f-structure evaluations in this chapter.

I use the pipeline parsing architecture shown in Figure 4.1 to generate c- and f-structures from raw strings taken from the ATIS corpus. I evaluate the c-structure trees output by the parser using PARSEVAL (Black et al., 1991) precision, recall and f-score metrics (calculated using `evalb`). The LDD-resolved f-structures output by the annotation algorithm in the pipeline parsing architecture are evaluated using the triple encoding and evaluation software of Crouch et al. (2002). Each f-structure is represented as a set of predicate-argument terms of the form

[1] I encountered some unresolvable robustness issues with retraining Charniak's parser on certain datasets. This has also been the experience of other researchers with whom I have discussed the issue.

[2] I am grateful to Aoife Cahill and Michael Burke for the help they gave me with this task.

`pred(arg1,arg2)` and precision and recall are calculated on these sets. The f-structure and corresponding triples for the sentence "Homer strangled Bart." are given below.

$$\left[\begin{array}{ll} \text{PRED} & \text{`STRANGLE}\langle(\uparrow\text{SUBJ})(\uparrow\text{OBJ})\rangle\text{'} \\ \text{SUBJ} & \left[\begin{array}{ll}\text{PRED} & \text{`HOMER'}\\ \text{NUM} & \text{SG}\\ \text{PERS} & 3\end{array}\right] \\ \text{OBJ} & \left[\begin{array}{ll}\text{PRED} & \text{`BART'}\\ \text{NUM} & \text{SG}\\ \text{PERS} & 3\end{array}\right] \\ \text{TENSE} & \text{PAST}\end{array}\right]$$

```
structure(
pers(bart~3,3)
obj(strangle~0,bart~3)
num(bart~3,sg)
pers(homer~2,3)
subj(strangle~0,homer~2)
tense(strangle~0,past)
num(homer~2,sg)
)
```

The CFG c-structure parser output is evaluated against the original 100 gold standard ATIS trees, and the f-structures are evaluated against the hand-crafted gold standard of f-structures for the 100 sentences from the ATIS corpus described above. I also carry out a CCG-style (Hockenmaier, 2003b) evaluation where I automatically generate f-structures for the entire ATIS corpus from the original ATIS treebank trees and evaluate f-structures generated from the parser output against these 578 "pseudo" gold standard f-structures in a 10-fold cross validation experiment.

Approximate randomisation statistical significance tests for the c-structures are performed using Bikel's script,[3] and for the f-structures the tests are performed using an approximate randomisation test program developed by Stefan Riezler (p.c.) which uses the output of the triple encoding and evaluation software of Crouch et al. (2002).

4.3.2 Experiments

I carried out two baseline evaluations. In the first evaluation (Table 4.1 (a)), I parse the raw strings for the 100 ATIS sentence gold standard with Bikel's parser (trained on Penn-II Treebank Sections 02-21) and pass the parser output to the automatic f-structure annotation algorithm of Cahill et al. (2004) to generate f-structures. I then evaluate the c-structure parse trees against the original treebank trees, and the automatically generated f-structures against the hand-crafted

[3] http://www.cis.upenn.edu/~dbikel/software.html

gold standard f-structures. In the second evaluation (Table 4.1 (b)), I parse the raw strings for the entire ATIS corpus (again with Bikel's parser trained on Penn-II Treebank Sections 02-21), and pass the parser output to the automatic f-structure annotation algorithm to generate f-structures. The parse trees are evaluated against the original ATIS treebank trees, and the f-structures are evaluated (CCG-style) against automatically generated f-structures for the 578 gold standard ATIS trees.

(a)

100 Gold Standard		Precision	Recall	F-Score
Trees (labelled bracketing)		73.77	67.05	70.25
F-Structures	All GFs	82.17	67.41	74.06
	Preds-only	70.33	56.97	62.95

(b)

578 ATIS		Precision	Recall	F-Score
Trees (labelled bracketing)		75.49	67.77	71.42[4]
F-Structures	All GFs	81.23	80.29	80.76
	Preds-only	69.27	67.02	68.13

(c)

DCU 105		Precision	Recall	F-Score
Trees (labelled bracketing)		86.56	85.59	86.07
F-Structures	All GFs	83.45	78.95	81.14
	Preds-Only	76.32	72.0	74.10

Table 4.1: Results for baseline experiments

[4] Note that the parsing result here for Bikel's parser using a grammar trained on WSJ Sections 02-21 of the Penn-II Treebank is slightly higher than that shown in Table 3.2 (f-score 69.94). This is because here Bikel's parser is used using slightly different settings which do not emulate Collins' Model 2 as closely as in the previous evaluation. The

Table 4.1 gives the results for the two experiments described above. Table 4.1 (a) shows the evaluation against the 100 sentence ATIS hand-crafted f-structure gold standard. Compared to the results for the Penn-II WSJ Section 23-based DCU 105[5] evaluation in Table 4.1 (c), the Penn-II Treebank-based LFG parsing resources of Cahill et al. (2004) show a significant drop in both the tree- and f-structure-based analysis scores for the ATIS material. The c-structures output by the parser show an f-score around 16% less than in the in-domain (Section 23-based) evaluation for the same parser (Bikel trained on Sections 02-21 of the Penn-II Treebank). Likewise the f-structure evaluation has suffered, with the preds-only f-score over 11% lower than on in-domain data. The CCG style evaluation in Table 4.1 (b), shows an all grammatical functions f-score of 80.76%, but a preds-only f-score of 68.13%.[6]

settings file `bikel.properties` distributed with Bikel's parser contains some extra settings for pruning, which do not appear in the settings file `collins.properties` which is used to make the parser emulate Collins' parser.

[5] http://nclt.dcu.ie/gold105.txt

[6] The CCG-style evaluation scores against the automatically generated f-structures for "perfect" treebank trees are difficult to compare with the hand-crafted DCU 105 gold-standard scores. The important comparison here is with the retraining results in Table 4.3

Dependency	Precision	Recall	F-Score
ADJUNCT	159/258=62	159/353=49	55
COMP	0/5=0	0/3=0	0
COORD	15/23=65	15/24=62	64
DET	56/64=88	56/70=80	84
FOCUS	**9/9=100**	**9/33=27**	**43**
OBJ	172/206=83	172/216=80	82
OBJ2	17/18=94	17/18=94	94
OBL	1/2=50	1/12=8	14
OBL2	0/0=0	0/5=0	0
POSS	1/1=100	1/1=100	100
QUANT	2/16=12	2/6=33	18
RELMOD	9/13=69	9/16=56	62
SUBJ	10/27=37	10/17=59	54
TOPICREL	**10/27=37**	**10/17=59**	**45**
XCOMP	23/33=70	23/46=50	58

Table 4.2: Dependency annotation results for selected features in the 100 sentence evaluation

Table 4.2 shows a more detailed analysis of the f-structure evaluation in Table 4.1 (a) for selected features. The table shows that in particular for features such as FOCUS, which indicates the role corresponding to the answer of the question, and TOPICREL, which indicates the relativised constituent in a relative clause, the performance is quite low. The FOCUS relation in particular is important to analyse correctly in questions (Harabagiu et al., 2000b).

While the precision for FOCUS relations is very high (100%), the recall is very low. Both precision and recall for TOPICREL relations are quite low. This indicates that, as it stands, the Penn-II Treebank-based LFG parsing system is not well suited to analysing questions and performance has suffered substantially as a result of the change in domain.

The drop in performance can be attributed to the domain variance, but the question remains as to which module in the pipeline parsing architecture in Figure 4.1 (c-structure parser, f-structure annotation algorithm or LDD resolution) is underperforming due to the change in domain. It may even be a combination of components? We can narrow the possibilities down to two of the three modules shown in Figure 4.1.[7] Either the c-structure parser is underperforming and consequently the annotation algorithm is unable to generate sufficiently good f-structures from the bad c-structures, or the annotation algorithm is incomplete with respect to the domain variance.

The results in Table 4.1 have shown that the c-structure parser performance has dropped by almost 16% as a result of the domain variance. Previous work (Gildea, 2001; Clark et al., 2004) and my experiments in Chapter 3 have shown that parser performance can be boosted through retraining with appropriate in-domain data. In order to to determine the influence of the c-structure parser as a contributing factor to the drop in performance for the automatic f-structure annotation algorithm, I carry out experiments using a c-structure parser retrained on Penn-II and ATIS material.

4.4 Retraining Experiments

In order to improve the performance of the c-structure parser on ATIS sentences I created an extended training set for the parser. This new, larger, training set consisted of Sections 02-21 of the Penn-II Treebank WSJ (the original training data) and the ATIS corpus less the 100 randomly selected sentences in the gold standard. I then retrained the parser on this new training set, and repeated the parsing and annotation experiments in Section 4.3.

[7] Testing on the long distance dependency resolution module showed that problems with LDD resolution were directly related to bad c-structure parsing. Manual inspection revealed that sentences with bad LDD resolution also had a bad c-structure tree, and that fixing the c-structure tree resulted in correct LDD resolution.

In the second (CCG-style) evaluation I generate c-structures for each of the 578 ATIS sentences by retraining the parser and parsing using a 10-fold cross-validation experiment with 90%:10% training:test splits over the ATIS corpus, and adding the 90% ATIS training set to Sections 02-21 of the Penn-II Treebank WSJ for training. Parser output for unseen data is generated for all 578 sentences in this way. The parser output c-structures are then passed to the f-structure annotation algorithm and LDD-resolution and the f-structures are evaluated against automatically generated f-structures from the original ATIS trees.

(a)

100 Gold Standard		Precision	Recall	F-Score	Diff
Trees (labelled bracketing)		88.03	78.78	83.14	+12.89
F-Structures	All GFs	88.04	79.10	83.33	+9.27
	Preds-only	80.17	73.66	76.77	+13.82

(b)

578 ATIS (Cross-validation)		Precision	Recall	F-Score	Diff
Trees (labelled bracketing)		80.66	92.26	86.07	+14.65
F-Structures	All GFs	87.27	88.97	88.11	+7.35
	Preds-only	80.21	80.81	80.51	+12.38

Table 4.3: Results for experiments with retrained grammar for the 100 sentence hand-crafted gold standard (a) and for CCG-style automatically generated gold-standard with a 10-fold cross validation (b).

Tables 4.3 (a) and (b) give the results of evaluating c-structures and f-structures generated with Bikel's parser retrained as described above. Evaluating against the 100-sentence ATIS gold standard, the c-structure f-score has increased by almost 13% to 83.14. The quality of the f-structures has also increased with an improvement of almost 14% in the preds-only f-score, to 76.77%. The performance over the whole corpus, in the CCG-style experiment against automatically generated f-structures for the original 578 treebank trees, has increased correspondingly, with the c-structure f-score increasing by over 14% to 86.07, and a preds-only evaluation of the f-structures gaining over 12% to achieve an f-score of 80.51.

The results in both evaluations show a considerable increase over the corresponding baseline scores in Table 4.1. This increase in performance of the c-structure parser and the automatic f-structure annotation algorithm is statistically significant for both evaluations with p-values of 9.999×10^{-5} for both c- and f-structures in the evaluation on the 100 sentence ATIS gold standard evaluation, and also on the CCG-style evaluation.

When compared with the baseline results for the Penn-II treebank-based DCU 105 in Table 4.1 (c), the improved results on the ATIS data (after retraining) are slightly better than the scores for the DCU 105. Both f-structure evaluations with both preds-only and all grammatical functions are higher for the 100 ATIS sentence gold standard after parser retraining than the DCU 105 result in Table 4.1. The cross-validation results for the whole ATIS corpus are higher again; though since the "gold-standard" for the evaluation was automatically generated (CCG-Style) for this test, it is not directly comparable with the DCU 105 result.

Dependency	Precision	Recall	F-Score	Diff
ADJUNCT	229/292=78	229/324=71	74	+19
COMP	0/4=0	0/3=0	0	-
COORD	16/24=67	16/24=67	67	+3
DET	67/66=92	61/70=87	90	+6
FOCUS	**23/23=100**	**23/33=70**	**82**	**+39**
OBJ	193/223=87	193/216=89	88	+6
OBJ2	17/17=100	17/18=94	97	+3
OBL	1/1=100	1/12=8	15	+1
OBL2	0/0=0	0/5=0	0	-
POSS	1/1=100	1/1=100	100	-
QUANT	2/16=12	2/6=33	18	-
RELMOD	14/19=74	14/16=88	80	+18
SUBJ	75/89=84	75/133=56	68	+14
TOPICREL	**14/19=74**	**14/17=82**	**78**	**+33**
XCOMP	25/30=83	25/46=54	66	+12

Table 4.4: Dependency annotation results for selected features in the 100 sentence evaluation

Table 4.4 shows a more detailed analysis of the evaluations in Table 4.3 (a) for a number of features. Compared to Table 4.2 the table shows that the retraining has had no negative effect on any of the features. The majority of features have improved in terms of both precision and recall.

The largest increase from the previous figures is for the features FOCUS and TOPICREL. These are important features necessary for analysing questions correctly. Note that recall for FOCUS has increased dramatically to 70% while precision has stayed the same at 100%. The

TOPICREL relation on the other hand gained substantially in terms of both precision and recall.

4.5 Further Evaluations and Backtesting

The experiments presented in this chapter and in Chapter 3 have shown that domain variation causes state-of-the-art treebank-based probabilistic resources such as the parsers of Collins (1999), Charniak (2000) and Bikel (2002) and the pipeline f-structure parsing architecture of Cahill et al. (2004) to under-perform resulting in poor quality c- and f-structure analyses. However, we have also seen that this underperformance can be remedied relatively easily through retraining on appropriate data from the new domain. In fact, only the c-structure parser needs to be retrained to improve the quality of analysis at both c- and f-structure level.

These findings raise a number of important research questions:

- What is the limit to which retraining can improve the results?
- How much training data is really needed?
- The ATIS corpus is a mixed corpus of question and non-question data, does this breakdown skew the results?
- What effect does retraining have on performance in the original domain (Penn-II Treebank)?

I address these questions in the sections below.

4.5.1 Upper Bound Estimation

The experiments so far indicate that the annotation algorithm of Cahill et al. (2004) is complete with respect to the strong domain variation encountered in the experiments on ATIS data. But this raises the question of what the upper bound for the automatic f-structure annotation of ATIS trees is.

In order to estimate an upper bound for the automatic f-structure annotation algorithm, I took the original ATIS treebank trees for the 100 sentences in the gold standard and automatically annotated them to produce f-structures (thereby removing the c-structure parser and f-structure-based LDD resolution margins of error). These f-structures were then evaluated against the hand-corrected f-structures in the gold standard. In this evaluation the all grammatical functions f-score is 92.80 and the preds-only f-score is 89.88 (Table 4.5). These results are roughly comparable to a similar estimation experiment on the Penn-II Treebank Section 23-based DCU 105.

	100 ATIS	DCU 105
	F-Score	F-Score
All GFs	92.80	96.93
Preds Only	89.88	94.28

Table 4.5: Upper bound for gold standard trees

The ATIS upper bound result reemphasises the earlier finding that improving the c-structure parsing is sufficient to improve the overall performance of the f-structure annotation algorithm significantly, and can be used to adapt the overall text-to-f-structure parsing system to data from outside of the domain on which it was originally developed. This is quite a surprising result, as the annotation algorithm of Cahill et al. (2004) was not modified in any way. The annotation principles used by the annotation algorithm were developed based on Penn-II Treebank data so the expectation was that the upper bound on out-of-domain data would be drastically lower than that from Penn-II Treebank data. One possible explanation for the surprising robustness of the f-structure annotation algorithm (given out-of-domain data but good c-structure analysis) may be that compared to c-structure, f-structure is a deeper, more abstract and "normalised" level of representation of linguistic information, less affected by domain variation. Even though there are very few direct questions in the original Penn-II data, the left-right context annotation matrices of the f-structure annotation algorithm (c.f. Section 2.3.3) contain sufficient information to annotate SBARQ and SQ constituents appropriately in question trees.

The upper bound results for the 100 ATIS sentence f-structure gold standard when compared with the results on the same test set with the retrained parser in Table 4.3 show that there is a further 9.47% f-score that can be gained in the all grammatical functions evaluation through improving parser output and another 13.11% f-score to be gained in the preds-only evaluation.

Speculating somewhat and taking the DCU 105 results as a yard stick, the results indicate that "perfect" c-structure parsing improves all grammatical functions f-score by approximately 9.5% and preds-only by approximately 13%. Part of the remaining 4% (approx.) difference between the ATIS and DCU 105 scores can potentially be bridged by improvements in the f-structure annotation algorithm to tune the algorithm to ATIS data. The upper bound tests indicate that the f-structure annotation algorithm is, in fact, less complete for ATIS data than for Penn-II material. One major factor is the high proportion of FRAG and X constituents in ATIS data for which the annotation algorithm currently does not provide annotations.

4.5.2 Ablation Experiments

We have seen above that adding a (relatively) small amount of domain appropriate material to the training set for the c-structure parser has resulted in quite significant gains for both c-structure and f-structure analyses of ATIS sentences. Previous work by Gildea (2001) has shown that a large amount of additional data makes little impact if it is not matched to the test material. With this in mind one can wonder if, due to its relative size, the Penn-II Treebank WSJ material in the training set for the parser might constitute such a large amount of redundant additional data.

In order to test this, I carried out a number of ablation experiments using the automatically f-structure annotated 578 ATIS trees as gold standard in a CCG-style experiment, where I evaluate c-structures and f-structures, while reducing the amount of Penn-II Treebank material in the parser's training set. The graphs in Figures 4.2 and 4.3 show the effect for evaluations against the entire ATIS corpus in a series of 10-fold cross validation experiments, in which the training set for the parser consists of 90% of the ATIS corpus and a varying (randomly selected) percentage of the Penn-II Treebank.

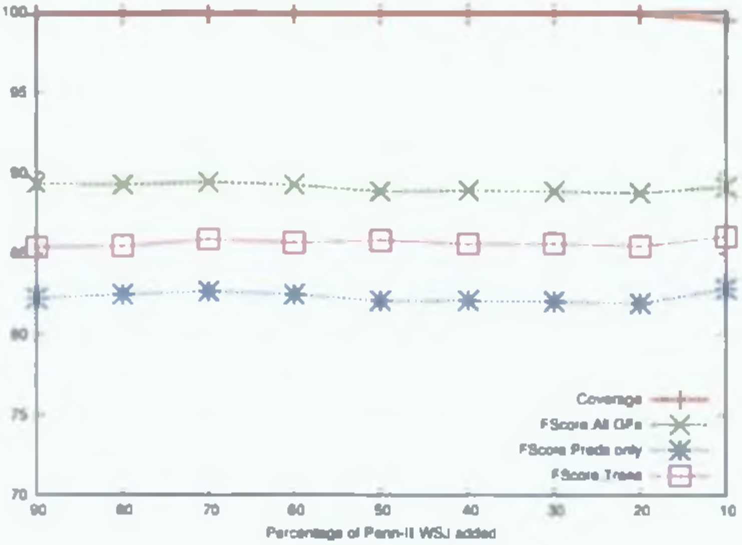


Figure 4.2: Reducing Penn-II Treebank content (90%-10% of sections 02-21 WSJ, 10-fold cross validation with 90%:10% ATIS splits, CCG-style experiment).

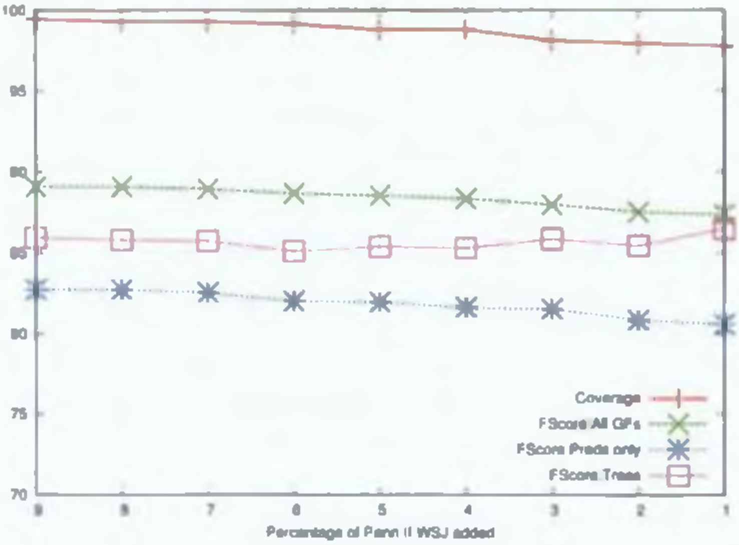


Figure 4.3: Reducing Penn-II Treebank content (9%-1% of sections 02-21 WSJ, 10-fold cross validation with 90%:10% ATIS splits, CCG-style experiment).

The graphs show that reducing the amount of Penn-II Treebank WSJ material in the training set adversely affects the overall performance. Grammar coverage, c-structure parsing and f-structure annotation all suffer to varying degrees. Both c-structure and f-structure evaluations start to decline when less than 70% of Sections 02-21 of the Penn-II Treebank is included in the training set. Grammar coverage proves to be less affected in this case: it does not decline until the amount of WSJ training material falls below 20%. Nevertheless, the system is capable of achieving coverage in the region of 99%, a c-structure f-score of over 85%, and f-structure f-scores of over 88% (all grammatical functions) and over 82% (preds-only), when the c-structure parser is trained on 90% of the ATIS corpus and only 10% of the Penn-II Treebank. It is interesting to note that the small variations seen in Figures 4.2 and 4.3 are for the most part only statistically significant for the c-structure parser. While the differences between adjacent runs start to become statistically significant for the c-structure parsing when the Penn-II Treebank material is below 70% of the whole, it is only when the Penn-II Treebank material is reduced below 10% that the differences in the f-structure evaluations are statistically significant.

These results show that while the additional Penn-II Treebank material is contributing only a small amount to the performance on ATIS material, the full set of Penn-II Treebank training material (Sections 02-21) is strictly speaking unnecessary and not making any significant contribution to the results. This is similar to what was observed in Chapter 3, where by and large c-structure parsing results did not change dramatically when Penn-II Treebank data was added to the ATIS training set. It also suggests that f-structure representations are less affected by the change in the training data than the c-structures, despite the fact that they are derived from them, implying that the automatic f-structure annotation algorithm is more robust and domain neutral than the c-structure parser.

4.5.3 Question vs. Non-Question Breakdown of Results

The ATIS corpus contains both question and non-question data. The 100-sentence gold standard is taken randomly from the ATIS corpus and comprises both question and non-question sentences. Table 4.6 shows the breakdown of the upper bound (established following the procedure detailed in Section 4.5) for both question and non-question sentences in the gold standard.

	Non-question	Question
All GFs	94.82	90.77
Preds-only	92.94	86.81

Table 4.6: Question and non-question f-score upper bounds

The upper bound breakdown shows a slight leaning towards a higher upper bound for non-question sentences, but the upper bound for questions is still quite high. To establish if there is a bias towards non-question data, I evaluated the question vs. non-question breakdown of the evaluations for parsing the 100 sentence gold standard before and after retraining on ATIS data.

Table 4.7 gives the breakdown of the scores for question and non-question sentences in the 100 sentence gold standard parsing evaluations.

	WSJ Trained		WSJ + ATIS Trained			
	Non-Question	Question	Non-Question		Question	
	F-Score	F-Score	F-Score	Diff	F-score	Diff
Trees	74.75	61.92	80.55	+5.8	88.35	+26.43
All GFs	77.40	70.52	82.62	+5.22	84.38	+13.86
Preds-only	68.96	54.12	76.28	+7.32	77.56	+23.44

Table 4.7: Question and non-question scores for the annotation algorithm

The breakdown in Table 4.7 clearly shows the effect of both the domain variance and the retraining in the earlier experiments. The left side of the table shows the breakdown for the baseline experiments before the parser was retrained. In this experiment it is clear that both the c-structure parser and the f-structure annotation algorithm are underperforming more on questions compared to non-question sentences. The right side of the table shows the same breakdown, but for the experiments with the parser retrained on both Penn-II Treebank WSJ and ATIS sentences. It is clear that this retraining has been of benefit to both the c-structure and f-structure evaluations for the questions in particular. The c-structure tree evaluation has improved significantly by over 26% with an f-score of 88.35, likewise the f-structure evaluations have improved for evaluations of all grammatical functions and preds-only, improving by 13.86% and 23.44% respectively. Significance testing on these results gives a p-value in the region of 9.999×10^{-5} for each evaluation. It is also interesting to note that none of the scores have decreased as a result of this retraining; the results for the non-question sentences have also improved significantly, albeit to a lesser extent (p-values in the region of 0.001).

The breakdown of results in Tables 4.6 and 4.7 shows that while the upper bound estimates in Table 4.6 show a higher result for non-question data in the test set, when the parser is retrained with a combination of Penn-II Treebank data and ATIS data the scores are higher for the question data than the non-question data. This is interesting as the upper bound estimates (Table 4.6) show the opposite result to what the retraining experiment shows. A possible explanation is that retraining with the ATIS data in the training set (while improving ATIS non-question scores) has a far greater beneficial effect on the question data than on the non-question data in the test set.

4.5.4 Backtesting

The experiments above show that retraining the c-structure parser for the new domain has allowed the treebank-based LFG resources of Cahill et al. (2004) to be adapted to a new domain and achieve similar f-scores in c- and f-structure evaluations on data from a new domain compared to in-domain results. In order to ensure that this retraining process has not adversely affected the overall system performance, I back-tested the retrained parser with the annotation algorithm on sentences from the original WSJ domain, the DCU 105 gold standard. I parsed the 105 sentences with each of the 10 retrained grammars from the 10-fold cross validation experiment in Section 4.4, then evaluated both c- and f-structures against the DCU 105 gold standard. The averaged results are shown in Table 4.8 (b), along with the results for the grammar trained only on sections 02-21 of the Penn-II Treebank in the same evaluation (a).

WSJ 02-21 trained		Precision	Recall	F-Score
Trees		86.56	85.59	86.07
F-Structures	All GFs	83.45	78.95	81.14
	Preds-Only	76.32	72.0	74.10

(a)

WSJ 02-21 + 90% ATIS trained		Precision	Recall	F-Score
Trees		87.05	86.10	86.57
F-Structures	All GFs	83.92	79.34	81.56
	Preds-Only	77.32	72.85	75.02

(b)

Table 4.8: Results for backtesting retrained grammar and baseline grammar on DCU 105

The results show that the retraining process has resulted in no loss of accuracy at either c- or f-structure level. The scores on the DCU 105 have in fact improved slightly as a result of the retraining; however the improvements, when tested, were not statistically significant. This indicates that there has been no significant negative effect on the LFG parsing resources of Cahill et al. (2004) on Penn-II WSJ material as a result of retraining the c-structure grammar to adapt the treebank-based LFG resources to a new domain.

Comparing the results in Tables 4.8 (b) and 4.3 (a) shows that although the retrained parser has a lower c-structure f-score for the 100 ATIS sentence test set than for the DCU 105 evaluation, the f-structure evaluation results are higher. This somewhat unusual result can perhaps be explained by the presence of relatively small errors in the c-structure trees, e.g. labelling SBARQ and SQ nodes incorrectly in questions, which do not affect the overall structure of the tree but which, because of the short sentence length in the ATIS data, can have a large impact on the labelled bracketing c-structure evaluation. These types of errors may sometimes be masked in f-structure, resulting in a better result in the f-structure evaluations. Further research is required to investigate this question.

4.6 Conclusion

This chapter has shown that similar to c-structure analysis, automatic f-structure annotation is adversely affected by domain variance. The pipeline model of Cahill et al. (2004) suffers a performance drop when tested on the ATIS corpus. Analysis of the results shows that features such as FOCUS, which Harabagiu et al. (2000b) identify as being very important to question analysis, and TOPICREL are not annotated very well by the baseline system.

Domain variance effects like this have been studied before by Gildea (2001) and Clark et al. (2004). This chapter presents work on deeper linguistic representations, f-structures, and how domain variance affects f-structure analysis as well as c-structure analysis. The experiments in Sections 4.3 and 4.4 show that both are adversely affected by domain variance, but that only the c-structure parser needs to be retrained to statistically significantly improve the performance of the overall system on ATIS data. This suggests that the more abstract f-structure representations are less subject to domain variation since the Penn-II Treebank-based automatic f-structure annotation algorithm needed no modification to cope with the new domain, once the c-structure parser had been retrained.

The experiments show that the Penn-II Treebank-based automatic f-structure annotation algorithm of Cahill et al. (2004) is general enough to cope with the domain variation observed here without modification: given high-quality c-structure trees, it can achieve a high upper bound for ATIS of 92.8% all grammatical functions and 89.88% preds-only for ATIS data, comparable to the upper bound for the automatic f-structure annotation algorithm on the Penn-II Treebank WSJ Section 23-based DCU 105. There is a provisio, however: given that the ATIS data contains many FRAG and X constituents, for which the f-structure annotation algorithm currently does not generate f-structures, there is more work to be done in order to properly handle this kind of data. The presence of FRAG and X constituents in the ATIS data may also explains the higher f-score result of 86.57% for c-structure trees for the retrained parser when tested on the DCU 105 versus 83.14% when tested on the ATIS 100 sentence gold standard.

Ablation experiments on the Penn-II Treebank WSJ component of the parser's training set show that, for tests on ATIS data, a large proportion of the Penn-II data is unnecessary and redundant in the parser's training corpus.

Investigating the question and non-question breakdown of the experimental results in this chapter shows that in the baseline evaluation the system suffers a worse drop in performance on question data than non-question data in both c-and f-structure evaluations. Retraining the c-structure parser with ATIS data to improve performance boosts the performance on both question and non-question data, giving a statistically significantly better result in both c-and f-structure evaluations of the whole system.

Importantly, experiments with the retrained (Penn-II + ATIS) grammar show that the retrained grammar, which gives high results for c- and f-structures for ATIS data, also produces good results when back-tested in the original domain. An evaluation of the retrained system against the Penn-II Treebank Section 23-based DCU 105 shows no negative effects on the c-structure parsing or f-structure annotations as a result of the retraining. The results are in fact higher than the original, though not statistically significantly so.

In summary, this chapter has built upon the domain variation experiments of Chapter 3, looking at the effects of domain variation on deeper linguistic representation in the form of LFG f-structures. The domain variation is evident at both c- and f-structure level particularly affecting the analysis of question specific dependencies such as FOCUS. To counter the drop in performance due to the domain variation only the c-structure parser needs retraining. This indicates that the automatic f-structure annotation algorithm of Cahill et al. (2004) is complete with respect to the domain variation observed here, and suggests that f-structures are a more abstract linguistic representation better able to cope with domain variance issues than c-structures. Adapting the f-structure pipeline parsing architecture of Cahill et al. (2004) in this way to cope with the new domain (ATIS) has no negative effects in the original domain.

Chapter 5

Creating a Question Treebank

5.1 Introduction

In Chapters 3 and 4, I have shown that retraining on a small amount of question data improves question analysis of both c- and f-structures. The upper bound estimations in Chapter 4 show that the existing retraining resources (the ATIS corpus) are insufficient in terms of both its size and content to achieve optimal results in question parsing. 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., 1994)). This can limit the applicability of grammatical resources induced from treebanks since such resources often underperform when used on a different type of text or for a specific task.

This chapter presents work on creating QuestionBank, a treebank of 4000 parse-annotated questions following Penn-II Treebank Annotation guidelines (Bies et al., 1995), which can be used as a supplementary training resource allowing 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. In either case, the resource will be useful in training parsers for use in question answering (QA) tasks, and also provide a means for evaluating the accuracy of these parsers on questions.

Section 5.2 provides the background and motivation for this work. The raw data sources for QuestionBank are described in Section 5.3. In Section 5.4, I describe the process of creating the question treebank and provide some statistics on the development of the corpus. Section 5.5 summarises and concludes.

Part of this work has been published in Judge et al. (2006).

5.2 Background and Motivation

High quality probabilistic, treebank-based parsing resources can be rapidly induced from appropriate training 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.

Work on retraining parsers to cope with domain variance was first studied by Gildea (2001) who retrained his parser with appropriate material to improve performance in the new domain. My experiments in Chapter 3 have shown that an even greater performance drop than that observed by Gildea occurs in instances of more extreme domain variance (Penn-II to ATIS). Clark et al. (2004) created a "What...?" question corpus for retraining a CCG supertagger which improves CCG parsing considerably. However, as it stands, the work is limited to one particular type of questions.

The question treebank described here is considerably larger than the "what" question corpus of Clark et al. (2004). It is intended to be used as a general training and evaluation resource for parsers used in the analysis of questions. In order to make it a representative question corpus, it contains a variety of question types (who, what, where, when, how, etc.) and is comprised of data from more than one source.

5.3 Data Sources

The raw question data for QuestionBank comes from two sources, the National Institute of Standards and Technology (NIST) Text Retrieval Conference (TREC) 8-11 QA track test sets,[1] and a question classifier training set[2] produced by the Cognitive Computation Group (CCG) at the University of Illinois at Urbana-Champaign. I used equal amounts of data (2000 questions) from each source so as not to bias the corpus to one or the other data source.

The questions in QuestionBank range between 2 and 32 words in length (ignoring punctuation) with an average length of around 8.5 words and a standard deviation of almost 3. A breakdown of the 4000 questions in QuestionBank by WH-word type is shown in Table 5.1.

Question Word	Frequency	Percentage of the whole
What	2294	57.35
How	480	12.00
Who	455	11.38
Where	243	6.08
When	200	5.00
Which	77	1.93
(Name)	62	1.55
Why	48	1.20
(Define)	3	0.08
(List)	2	0.05
Other	136	3.40

Table 5.1: Breakdown of the 4000 Questions in QuestionBank by WH-word

The 136 questions in the "Other" category in Table 5.1 fall into three general categories,

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

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

examples are listed below. They are either inverted questions (1), WHPP fronted questions (2), NP topicalised questions (3), or other imperatives (4) and (5) not listed in Table 5.1.

1. Logan International serves what city?
2. On which Hawaiian island is Pearl Harbour?
3. The sun's core, what is the temperature?
4. Describe the Finnish music personality Salonen's appearance.
5. Tell me what city the Kentucky Horse Park is near?

5.3.1 TREC Questions

The TREC competitions (Voorhees, 2001) have become the standard evaluation for QA systems. TREC QA test sets consist primarily of fact seeking questions with some imperatives which request information, e.g. "List the names of cell phone manufacturers." In the corpus I have included 2000 TREC questions, the test questions for the 1999, 2000, 2001 and 2002 TREC QA track (totaling 1893 questions) and 107 questions from the 2003 TREC QA track test set.

The first TREC QA test questions (1999) came from a four sources, TREC QA participants, the NIST TREC team, the NIST assessors and question logs from the FAQ Finder system (Burke et al., 1997). The different sources provided different kinds of questions as each source had their own motivation for the questions asked. The TREC QA participants and NIST staff (experts on QA) were able to select "interesting" and challenging questions, the assessors represent a general user's point of view and the FAQ Finder logs represent real questions posed to a QA system. Subsequent test sets for TREC QA tasks have been created in a similar manner, sourcing questions from domain experts and real life query logs.

5.3.2 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 (Li and Roth, 2002). The 5500 questions were stripped of answer type annotation, duplicate questions (both within the 5500 and duplicates of TREC questions) were removed and 2000 questions were included in the question treebank.

The CCG questions are classified according to a two-tier classification, a coarse grained classification (e.g. LOCATION) is assigned, as well as a fine grained classification (e.g. CITY), resulting in a total of 50 classes. A breakdown of the CCG question data by coarse classes is given in Table 5.2.

Class	Frequency	Percentage of the whole
Abbreviation	36	1.8
Entity	376	18.8
Description	552	27.6
Human	260	13
Location	324	16.2
Numeric	452	22.6

Table 5.2: Breakdown of 2000 CCG questions by question class

Similar to the TREC questions, the 5500 CCG questions come from a number of sources 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 carried out (e.g. separating contractions), but the minor grammatical errors were left unchanged because I 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.

5.4 Creating QuestionBank

I used a bootstrapping method to create the question treebank. As the treebank grows, the parser trained on the growing treebank becomes more accurate and the human annotator(s) should need to do progressively less manual correction work and the whole process speeds up.

5.4.1 Bootstrapping a Question Treebank

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

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 of the data has been parsed and hand corrected. The parser used to process the raw questions prior to manual correction was that of Bikel (2002)[3], 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. I 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.

[3] Downloaded from Daniel Bikel's website at the University of Pennsylvania, http://www.cis.upenn.edu/~dbikel/software.html#stat-parser

In the first iteration of the bootstrapping algorithm 200 questions were parsed. This increased over the course of the development of the treebank; the final iteration ran with 500 questions being parsed.

5.4.2 Corpus Development Statistics

QuestionBank 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, e.g. a corpus of a similar number of Spanish sentences (Civit and Martí, 2004) took approximately 15 months to create. The speed of the process was helped by three main factors: unlike the Cast3LB Spanish treebank, I used existing annotation guidelines (Bies et al., 1995) instead of developing my own treebank annotation; the questions are generally quite short (typically about 10 words long) compared to sentences in general newspaper text (Table 3.1) 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.

5.4.3 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 (Figure 5.1).

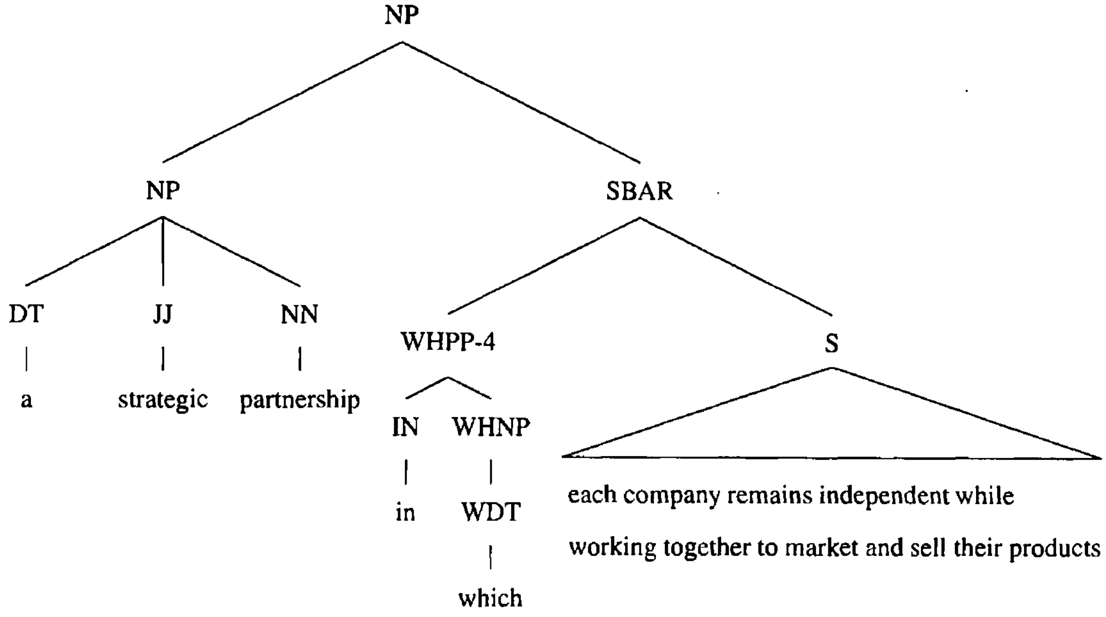

NP
NP
SBAR
DT
JJ
NN
a
strategic
partnership
WHPP-4
S
IN
WHNP
in
WDT
which
each company remains independent while working together to market and sell their products

Figure 5.1: Relative clause in an NP from the Penn-II Treebank

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) as in Figure 5.2.

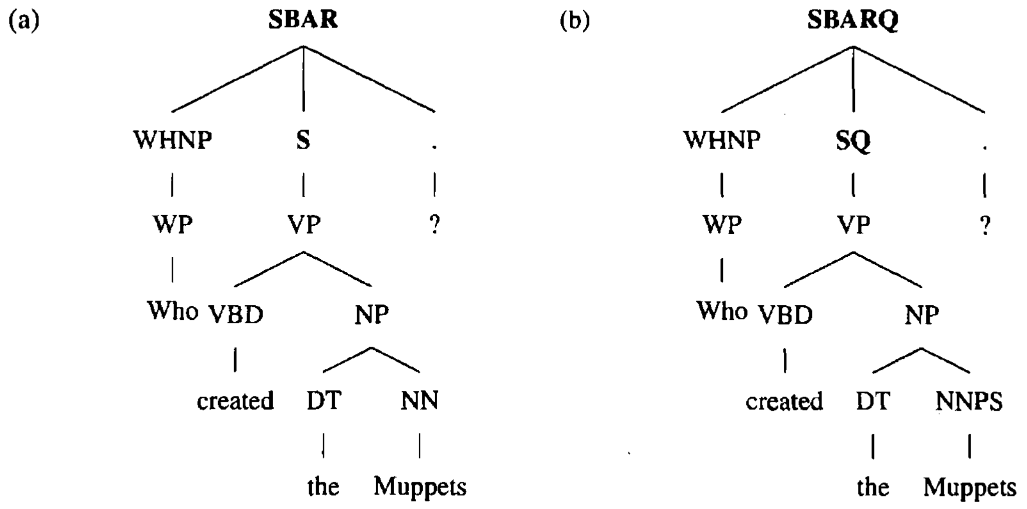


Figure 5.2: Example tree before (a) and after correction (b)

Because the questions are typically short, an error like this has quite a large effect on the PARSEVAL (Black et al., 1991) score for the overall tree in terms of labelled precision, recall and f-score as explained in Chapter 3, Section 3.4.1. In Figure 5.2, the f-score for the parser output (a) would only be 60% if it was evaluated against the tree in (b). 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. 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 5.3(a), where the PP mother of the WHNP is not correctly labelled as a WHPP as in Figure 5.3(b)

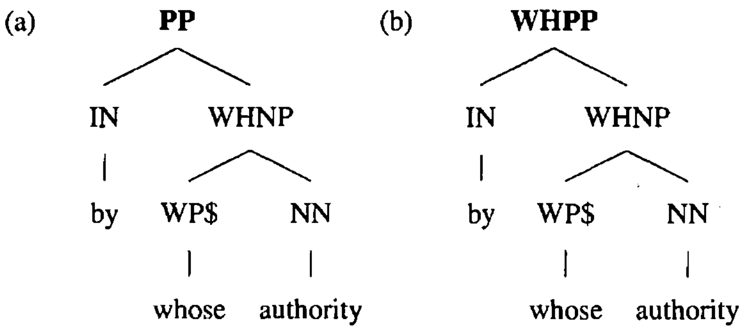


Figure 5.3: WH-XP assignment before (a) and after correction(b)

The parser output often had to be rearranged structurally to varying degrees. This was common in 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 5.4(a) illustrates this, and Figure 5.4(b) shows the corrected tree with the VP node inserted.

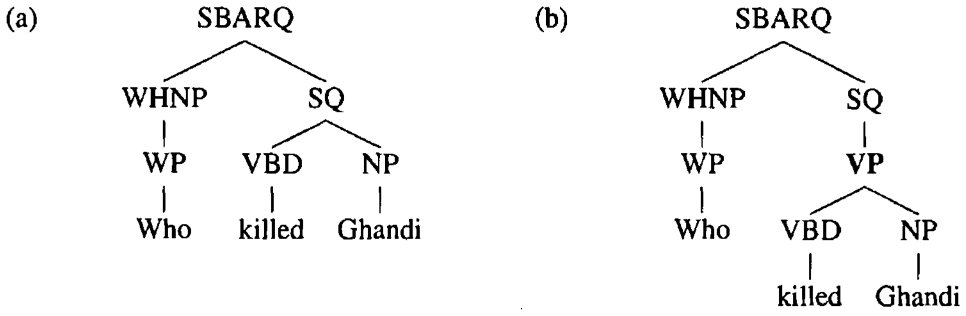


Figure 5.4: VP missing inside SQ with a single object NP (a) and correction (b)

On inspection, I found that the problem was caused by copular interrogative 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 5.5(a)).

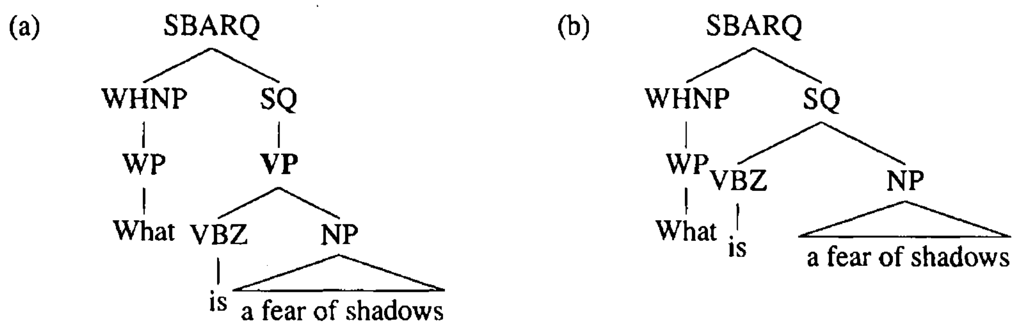


Figure 5.5: Erroneous VP in copular construction (a) and correction (b)

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.

5.5 Summary

In this chapter I have presented work on the creation of a corpus of 4000 parse annotated questions, QuestionBank. QuestionBank was created as a training and evaluation resource for probabilistic and machine learning treebank-based parsers. The raw data used to create QuestionBank came from QA test sets (TREC) as well as more varied sources (CCG question classifier corpus) to give a representative question corpus.

In Chapter 6, I present a series of experiments to test the effectiveness of QuestionBank for the task of training a parser to analyse question data and also test retrained parsers on non-question data in WSJ Section 23 of the Penn-II Treebank. Chapter 7 presents work on completing QuestionBank by adding WH-traces to the trees.

Chapter 6

Experiments with QuestionBank

6.1 Introduction

In this chapter I present a number of experiments to test the effects of training a parser on QuestionBank versus the Penn-II Treebank, and on a combination of the two. The aim of the experiments is (i) to test whether training on QuestionBank can lead to an improvement in question parsing; (ii) to test if training on QuestionBank has negative effects on the parser's ability to parse non-question data (WSJ Section 23 of the Penn-II Treebank); (iii) to investigate if QuestionBank is sufficiently large to be an effective training/evaluation resource for questions.

Section 6.2 briefly outlines the experiments. Section 6.3 describes the data used for the experiments, the experiments themselves and the results and Section 6.4 summarises and concludes.

The space of possibilities for comparing training and test sets evaluated in the baseline and cross validation experiments explored in this chapter is shown in the table below.

Test Training	Penn Sect 23	QuestionBank (all)	QB Cross Validation
Penn	*	*	
QB Cross Validation	*		*
Penn + QB	*		*

Here training on Penn refers to training on Sections 2-21 of the Penn-II Treebank, QB Cross Validation to a 10-fold cross-validation training and testing on all of QuestionBank and Penn + QB to a 10-fold cross validation testing on QuestionBank and training on QuestionBank plus all of Sections 2-21 of the Penn-II Treebank.

Part of this work has been published in Judge et al. (2006).

6.2 Outline of Experiments

I carried out seven experiments in total, a baseline experiment, two cross-validation experiments and five ablation experiments. In the cross-validation experiments I use all 4000 trees in the completed QuestionBank as the test set. Ten-fold cross validation on the 4000 questions was carried out using 90%/10% splits of the data, with 90% training material and 10% test set. In this way parser output for unseen data was generated for all 4000 questions in QuestionBank. In the first cross-validation experiment, **Cross-Validation 1,** I use only QuestionBank as a training resource, in the second, **Cross-Validation 2,** I use QuestionBank and Sections 02-21 of the Penn-II Treebank.

I 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 I used QuestionBank and Penn-II Treebank training material for the parser. The first ablation experiment, **Ablation 1,** uses only QuestionBank, the second, third and fourth ablation experiments, **Ablation 2, Ablation 3** and **Ablation 4,** use QuestionBank and Sections 02-21 of the Penn-II Treebank, **Ablation 5** uses only Sections 02-21 of the Penn-II Treebank.

In all of the experiments I backtest the retrained parser on Section 23 of the Penn-II Treebank to compare the performance on questions as well as on informative text.

6.3 Experimental Data

For the cross-validation experiments it was possible to use all 4000 questions in QuestionBank as a gold standard against which to evaluate parser output. However, for the ablation experiments this is not possible because a consistent test set is needed if the results from the different runs are to be comparable. For the ablation experiments the 4000 questions were divided into two sets. 400 randomly selected trees were held out as a gold standard test set against which to evaluate, the remaining 3600 trees were used as a training corpus.

The Penn-II Treebank material used for both the cross-validation and ablation experiments was the standard training/test sets: WSJ Sections 02-21 were used as the training set and WSJ Section 23 was used as the test set.

6.3.1 Establishing the Baseline

The baseline for my experiments is provided by Bikel's parser trained on WSJ Sections 02-21 of the Penn-II Treebank. I test on all 4000 questions in QuestionBank, and also Section 23 of the Penn-II Treebank.

	QuestionBank	WSJ Section 23
Coverage (%)	100	100
F-Score (%)	78.77	82.97

Table 6.1: Baseline parsing results

Table 6.1 shows the results for the baseline evaluations on both the question and non-question test sets. While the coverage for both tests is high (100%), the parser underperforms considerably 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. This is a higher baseline result than in the ATIS evaluations (Table 3.2), and is most likely because of the higher proportion of grammatically complete sentences in QuestionBank, with only 10 sentences containing FRAG constituents in QuestionBank compared to 230 in ATIS. It is doubtful that sentence length plays any part in the difference in baseline scores between the two test sets as the average sentence length for QuestionBank is 8.5 words, and 7 words for ATIS. Note that in my evaluations I *include punctuation*, which is not included in the published results for Bikel's parser.[1] I include punctuation in my evaluations as punctuation, e.g. commas, helps determine the structure of a sentence. The parsing model of Collins (1999) places very little importance on punctuation, attaching it at a point in the tree that places it between two non-terminals; Bikel (2002) similarly places little importance on punctuation, attaching it as high in the tree as is possible (Figure B.1 in Appendix B).

6.3.2 Cross-Validation Experiments

I carried out two cross-validation experiments to investigate the accuracy of a parser trained on QuestionBank, and on QuestionBank plus Sections 02-21 of the Penn-II Treebank, when parsing question data.

Cross-Validation 1: In the first experiment I perform a 10-fold cross-validation experiment using the 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 and the parser was trained on the remaining 90% of QuestionBank (only). In this way parses for unseen data were generated for all 4000 questions and evaluated against the QuestionBank trees. Each of the ten grammars is also back-tested on Section 23 of the Penn-II Treebank and the average scores are reported.

[1] Results for Bikel's parser in the literature are often those for the parser when emulating Collins' Model 2. This mode of operation includes specific optimisations for the Penn Treebank and pruning, which I do not use here. Also when evaluating parser output, Bikel, following Collins ignores punctuation. In my evaluations I do not ignore puntuation. The same evaluation carried out using the Collins Model 2 emulation settings file supplied with Bikel's parser gives f-scores of 81.77% for the 4000 QuestionBank questions and 86.23% for Section 23.

	QuestionBank	WSJ Section 23
Coverage (%)	100	98.79
F-Score (%)	88.82	59.79
p Value (Baseline)	9.999×10^{-5}	9.999×10^{-5}

Table 6.2: Experiment **Cross-Validation 1**. Cross-validation experiment using the 4000 question treebank

Table 6.2 shows the results for the first cross-validation experiment (**Cross-Validation 1**), using only the 4000 sentence QuestionBank. Compared to Table 6.1 the results show a statistically significant (p-value of 9.999×10^{-5}) improvement of over 10% on the baseline f-score for questions. However, the tests on the Section 23 data show not only a statistically significant (p-value of 9.999×10^{-5}) drop in f-score from 82.97% to 59.79% but also a drop in coverage from 100% to 98.79%. So while the parser can achieve high accuracy and coverage on question data, its ability to parse Section 23 data has suffered significantly.

Cross-Validation 2: The second cross-validation experiment was similar to the first, but in each of the 10 folds I train the parser on 90% of the 4000 questions in QuestionBank and on all of Sections 02-21 of the Penn-II Treebank. Again each of the extracted grammars is tested against the 10% of QuestionBank held out, and against Section 23 of the Penn-II Treebank, and average scores are reported.

Questions		Backtest on Sect 23	
Coverage (%)	100	Coverage (%)	100
F-Score (%)	89.75	F-Score (%)	82.39
p Value (Baseline)	9.999×10^{-5}	p Value (Baseline)	0.0002

Table 6.3: Experiment **Cross-Validation 2**. Cross-validation experiment using Penn-II Treebank Sections 02-21 and 4000 questions

Table 6.3 shows the results for the second cross-validation experiment (**Cross-Validation 2**) 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 6.2). The non-question results are also better and are comparable to the baseline (Table 6.1), though there is a 0.58% drop in f-score, which is statistically significant. Including the Penn-II Treebank data with QuestionBank gives better scores for both question and non-question data than training on QuestionBank alone.

If we compare these results to those in Chapter 3 (where the finding was that for Bikel's parser when testing on an ATIS test set, adding Penn-II Treebank data had a very small effect on the performance on ATIS data), we find that for the QuestionBank tests combining the training sets (QuestionBank + PTB) has resulted in a higher score than training on either resource alone. However, for tests on Penn-II Treebank Section 23, combining the training sets results in a higher score than when trained on QuestionBank alone, but a slightly lower result than when trained on Penn-II Treebank data alone.

The higher f-score (89.75%) on the QuestionBank data than on Section 23 suggests that with sufficient appropriate retraining data, the parser can achieve higher results on the question data than on informative text. This is not entirely unsurprising since the questions are generally much shorter than informative sentences, and as the graphs in Figures 3.7 and 3.8 show, state-of-the-art parsers are capable of very high f-scores on sentences of short lengths. This suggests that the low baseline score on the question data is not due to the sentence length, but rather to poor performance on question specific constructions, which can be remedied through retraining.

Questions

	Baseline	Cross-Validation 1	Cross-Validation 2
Baseline	-	-	-
Cross-Validation 1	9.999×10^{-5}	-	-
Cross-Validation 2	9.999×10^{-5}	0.0002	-

Section 23

	Baseline	Cross-Validation 1	Cross-Validation 2
Baseline	-	-	-
Cross-Validation 1	9.999×10^{-5}	-	-
Cross-Validation 2	0.0002	9.999×10^{-5}	-

Table 6.4: Statistical significance comparison between cross-validation runs

Table 6.4 shows the statistical significance between the baseline evaluation and each of the cross-validation runs and also between the cross-validation runs. The results show that the differences between each of the evaluations are all statistically significant. The improvement in question parsing in experiment **Cross-Validation 1** over the baseline is significant, as is the improvement in question parsing from **Cross-Validation 1** to **Cross-Validation 2**. The drop in performance on Section 23 from the baseline to both **Cross-Validation 1** and **Cross-Validation 2** is also significant.

6.3.3 Ablation Runs

In a further set of experiments I investigated the effect of varying the amount of data in the parser's training corpus. I experimented with varying both the amount of QuestionBank and

Penn-II Treebank data that the parser is trained on. In each experiment, I test on both the 400 question test set and Section 23 of the Penn-II Treebank.

Ablation 1: In the first experiment, I train the parser using only the 3600 question training set. I 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. Table 6.5 gives the results for this experiment.

% Question Material		100	90	80	70	60	50	40	30	20	10
Coverage	Questions	100	100	100	100	100	100	100	100	100	100
	Section 23	98.55	99.13	99.17	97.89	99.34	99.17	99.59	99.05	97.97	98.84
F-Score	Questions	89.24[2]	89.13	88.9	88.36	87.87	88.56	87.52	87.41	87.08	85.59
	Section 23	59.61[2]	58.82	58.44	58.26	56.87	55.87	55.06	53.4	52.8	51.85

Table 6.5: Results table for experiment **Ablation 1**

[2] The f-score results in this column differ from the results in Table 6.2 because the ablation experiments are tested on a single 400 question test set, while Table 6.2 gives 90%/10% split-based cross validation results on the whole of the 4000 questions in QuestionBank.

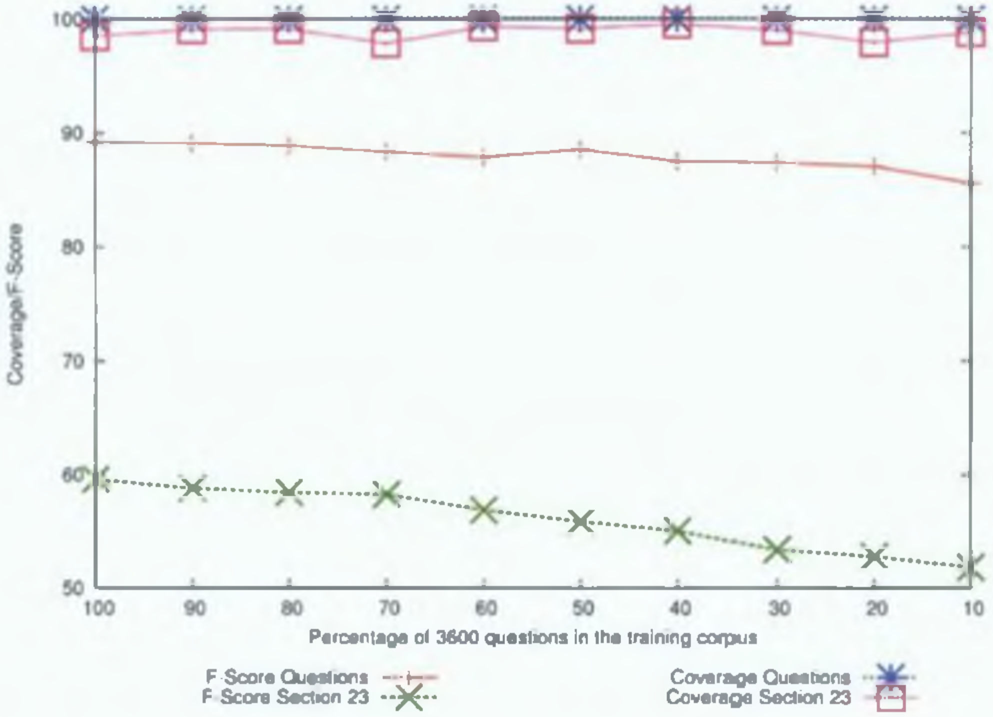


Figure 6.1: **Ablation 1**: Results for ablation experiment reducing 3600 training questions in steps of 10%

Figure 6.1 graphs the coverage and f-score for the parser for 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 a high degree of 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 a labelled precision and recall 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, the most it accomplishes is 99.59% of the test set, and the best precision and recall f-score at 59.61% is very low.

Ablation 2: The second experiment is similar to the first, but in each run I add Sections 02-21 of the Penn-II Treebank to the (shrinking) training set of questions. Table 6.6 gives the results for this experiment.

% Question Material		100	90	80	70	60	50	40	30	20	10
Coverage	Questions	100	100	100	100	100	100	100	100	100	100
	Section 23	100	100	100	100	100	100	100	100	100	100
F-Score	Questions	91.06[1]	91.39	91.29	91.29	91.01	90.93	90.9	90.63	89.97	88.91
	Section 23	82.38[1]	82.36	82.38	82.42	82.41	82.41	82.43	82.39	82.42	82.46

Table 6.6: Results for experiment **Ablation 2**

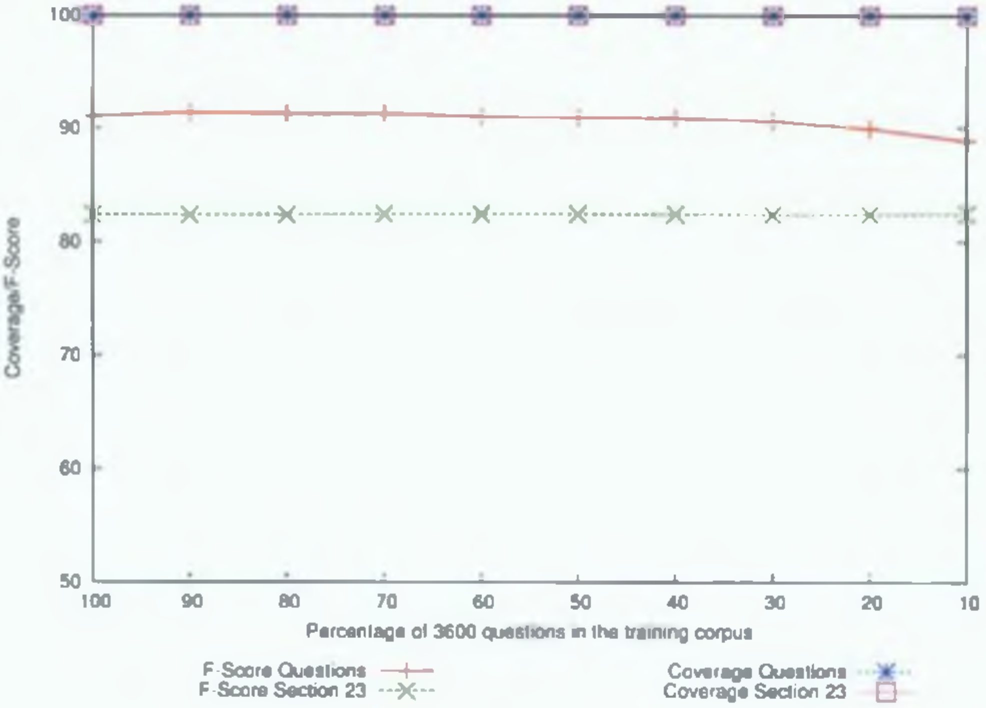


Figure 6.2: **Ablation 2**: Results for ablation experiment using PTB Sections 02-21 (fixed) and reducing 3600 questions in steps of 10%

Figure 6.2 graphs the results for the second ablation experiment as described above. The

[1] The f-score results in this column differ from the results in Table 6.3 because the ablation experiments are tested on a single 400 question test set, while Table 6.3 gives 90%/10% split-based cross validation results on the whole of the 4000 questions in QuestionBank.

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 parser is tested on both the 400 question test set, and Penn-II Treebank 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 of 82.97.

Ablation 3: 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. Table 6.7 gives the results for this experiment

% Penn-II Material		100	90	80	70	60	50	40	30	20	10
Coverage	Questions	100	100	100	100	100	100	100	100	100	100
	Section 23	100	100	100	100	100	100	100	100	100	100
FScore	Questions	91.06	91.03	91.44	91.31	91.26	91.29	91.08	91.15	90.71	90.3
	Section 23	82.38	82.31	82.24	82.01	82.07	81.5	81.32	80.8	80.48	79.69

Table 6.7: Results for experiment **Ablation 3**

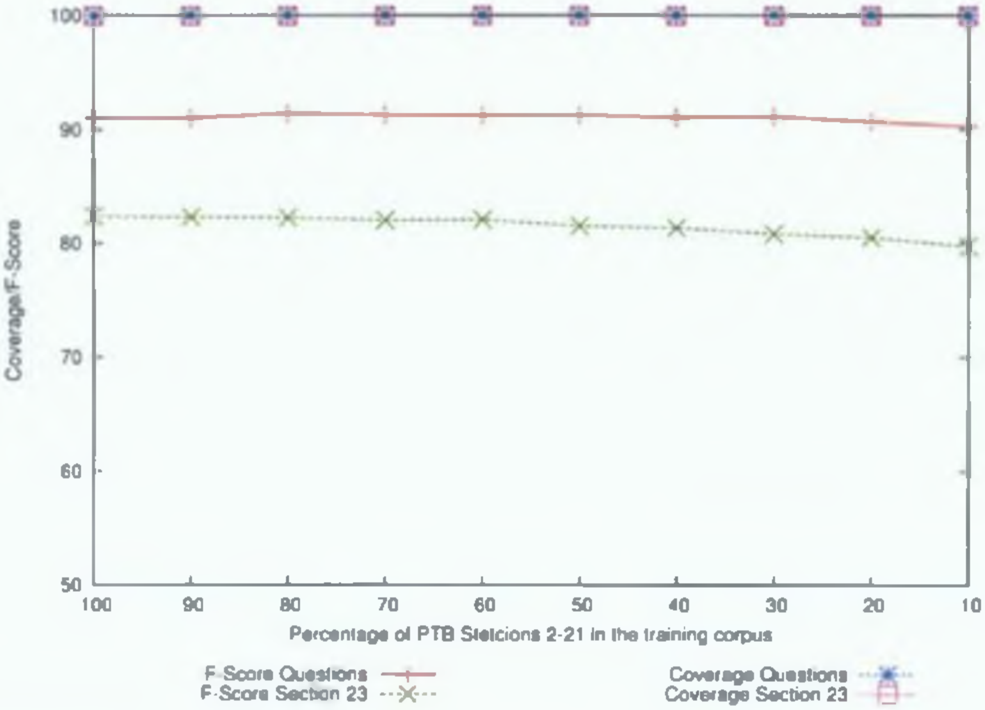


Figure 6.3: **Ablation 3**: Results for ablation experiment using 3600 questions (fixed) and reducing PTB Sections 02-21 in steps of 10%

Figure 6.3 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-II 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.

Ablation 4: In the fourth ablation experiment I reduce both the amount of Penn-II Treebank and QuestionBank material in the parser's training corpus. The amount of data from each source in the training corpus is reduced by 10% each time. For the first run, the training set consists of 100% of Sections 02-21 of the Penn-II Treebank and 100% of the 3600 QuestionBank training corpus. For the second run the training set consists of 90% of Sections 02-21 of the Penn-II

Treebank and 90% of the 3600 QuestionBank training corpus, and so on. Again I test on the 400 question test set and on Section 23 of the Penn-II Treebank. Table 6.8 gives the results for this experiment.

		100	90	80	70	60	50	40	30	20	10
Coverage	Questions	100	100	100	100	100	100	100	100	100	100
	Section 23	100	100	100	100	100	100	100	100	100	100
FScore	Section 23	91.06	91.37	91.49	91.15	90.66	90.97	90.51	89.90	88.99	88.84
	Questions	82.38	82.32	82.30	82.00	82.02	81.57	81.41	80.84	80.58	79.68

Table 6.8: **Ablation 4:** Results for ablation experiment using 3600 questions and PTB Sections 02-21, reducing the amount of material from *both* sources in steps of 10%

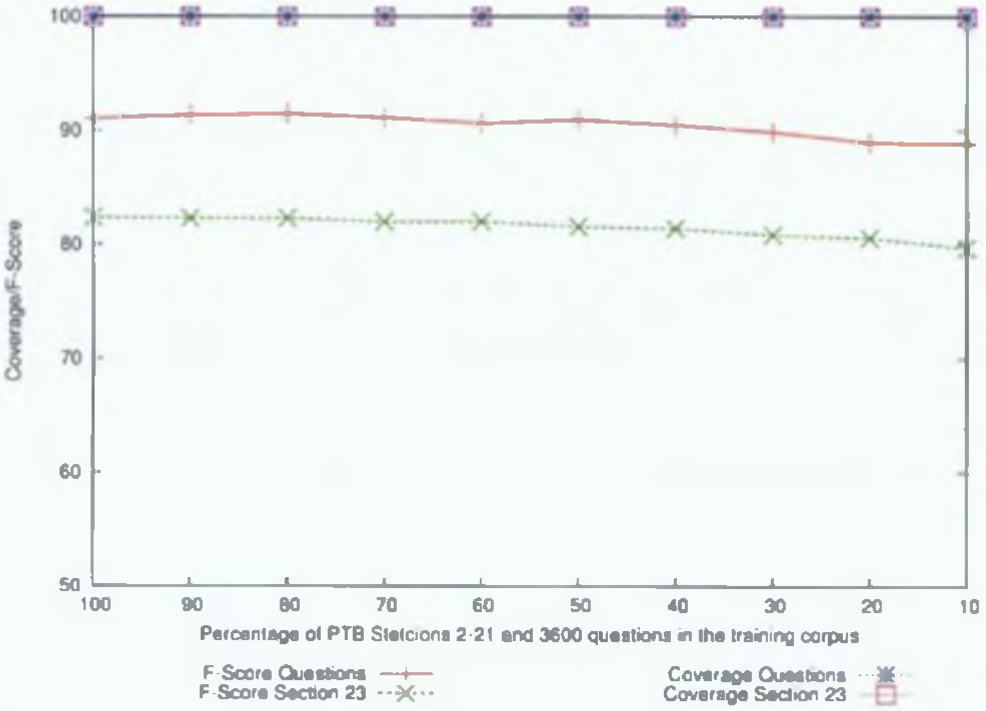


Figure 6.4: **Ablation 4:** results for ablation experiment using 3600 questions and PTB Sections 02-21 reducing each in steps of 10%

Figure 6.4 shows the results for the fourth ablation experiment. Here the amount of material from *both* corpora (QuestionBank and Penn-II Treebank) is reduced. The parser's performance on both test sets is affected in a similar way as the amount of data in the training corpus is reduced. The (red) graph line representing f-score for the question test set in Figure 6.4 is similar in shape to the corresponding line in Figure 6.2 for experiment **Ablation 2** where the parser is trained on a reducing amount of question material added to a fixed amount of Penn-II Treebank material (Sections 02-21). The results in Table 6.6 and 6.8 show that there is little difference in the question f-scores for both runs, the variation between the corresponding runs is not statistically significant with an average p-value of 0.306. Similarly, the (green) graph line representing f-score on Section 23 of the Penn-II Treebank in Figures 6.3 and 6.4 follow the same trend, and the data in Tables 6.7 and 6.8 show there is little difference between the figures. Again, the variation is not statistically significant with an average p-value of 0.276.

Ablation 5: In the final ablation experiment I carried out, the parser's training corpus consists of reducing amounts of data from Sections 02-21 of the Penn-II Treebank. No question data was included in the training corpus. The results for this experiment are in Table 6.9

		100	90	80	70	60	50	40	30	20	10
Coverage	Questions	100	100	100	100	100	100	100	100	100	100
	Section 23	100	100	100	100	100	100	100	100	100	100
FScore	Questions	78.42	79.00	79.31	78.97	75.87	74.98	73.42	73.63	68.62	72.55
	Section 23	82.97	82.45	82.26	82.01	82.06	81.58	81.43	80.96	80.56	79.57

Table 6.9: Results for experiment **Ablation 5**

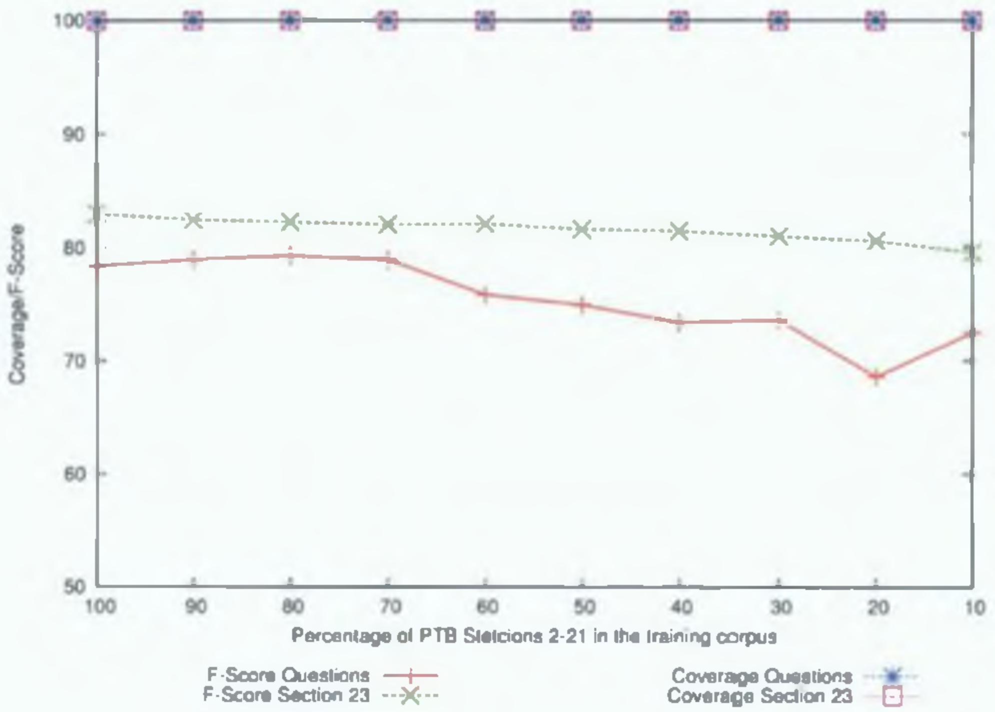


Figure 6.5: **Ablation 5:** Results for ablation experiment reducing the amount of training data from Sections 02-21 of the Penn-II Treebank by 10%

Figure 6.5 shows the results for the fifth ablation experiment, **Ablation 5**, the graph shows that as the amount of Penn-II Treebank material in the parser's training set is reduced the performance on both Section 23 and the question test set steadily declines. The performance on questions declines to a greater degree than the performance on Section 23.

6.4 Summary

The experiments reported in this chapter have shown that a parser trained exclusively on QuestionBank is capable of parsing questions more accurately than if trained on the Penn-II Treebank alone. However, the accuracy on a non-question test set (Section 23 of the Penn-II Treebank) is poor. Interestingly, when we train the parser on a combination of Penn-II Treebank and QuestionBank data, the parse accuracy on questions increases and the parser is still able to parse a non-question test set with state-of-the-art accuracy.

The research shows that QuestionBank, if used exclusively, can be used to extract a parser which is significantly smaller and faster[4] than one extracted from general training corpora, like the Penn-II Treebank, and which can analyse questions with a high degree of accuracy. Such capabilities are desirable if the parser is to be used in a QA system exclusively for query analysis. QuestionBank can also be used in conjunction with the Penn-II Treebank, to train a parser which can analyse questions with a high degree of accuracy (89.75% labelled precision and recall f-score), and do the same for non-question, informative text like that found in the WSJ data in the Penn-II Treebank (82.39% labelled precision and recall f-score). These are qualities that are required of a parser if it is to be used in a QA system where it will be expected to analyse both queries and potential answers.

Having noted the significant improvement in question parsing accomplished with a relatively small question corpus (4000 questions in QuestionBank vs. $\approx$ 40,000 sentences in the Penn-II Treebank training sections), an interesting question is whether more can be gained by increasing the size of the question corpus, or whether the amount of training material in QuestionBank constitutes an upper bound. Analysing the graph in Figure 6.1 which displays the effect of reducing the amount of questions in the training set, shows that the leftmost portion of the curve for f-score for the question test set is relatively flat in the region corresponding to 50-100% of the question training data. Relative to the amount of additional question training data there is little change in the f-score in this region: the f-score at 50% of the question training data is 88.56 and at 100% of the question data it is 89.24, an increase of less than 0.7 of a percent which is not statistically significant (p-value of 0.14). It is interesting to note that in this region the f-score for the non-question parsing increases by a much greater degree (3.74%) and that this increase *is* statistically significant (p-value of 9.999×10^{-5}). This implies that while I have not found

[4] Basic benchmarking tests performed on Section 23 of the Penn-II Treebank, and the 400 question test set described in Section 6.3 show that on average for Bikel's parser trained on the 3600 question training set parses Section 23 approximately 3 times faster and the 400 questions approximately 8 times faster than a grammar trained on Sections 02-21 of the Penn-II Treebank.

an absolute upper bound, the question corpus is sufficiently large that the gain in accuracy from adding more question data is so small that it does not justify the effort.

QuestionBank is sufficiently large to fulfill the task for which it was intended, to provide a training and evaluation resource for question parsing, and to establish an upper bound on the amount of additional data required in the training corpus to have a worthwhile gain in parser accuracy.

Extrapolating the findings in Chapter 4 that improved c-structure analysis results in improved f-structure output from the automatic f-structure annotation algorithm of Cahill et al. (2004), the results of experiments in this chapter would suggest that the improvements in c-structure analysis of questions from a parser trained using QuestionBank would carry through into f-structure analysis. I hope to be able to pursue this in future work.

Chapter 7

Adding Long Distance Dependency Information to QuestionBank

7.1 Introduction

This chapter presents work on adding long distance dependencies (LDDs) to QuestionBank. After "bootstrapping" the treebank from raw data, the trees did not contain information on long distance dependencies (WH-traces and coindexation etc.) because the parser does not output this information. A number of methods exist for recovering this information from parser output. I present a new high-precision method to recover LDDs in parser output using reentrancies in automatically generated LDD resolved f-structures to "reverse engineer" the syntactic components (i.e. the trace and its coindexed antecedent) in the tree. I evaluate this method of recovering LDDs against questions and their gold standard syntax trees (with LDDs indicated in terms of empty productions and coindexed antecedents) from the ATIS corpus and, where possible, compare the approach with the other methods discussed in Section 7.3.

Long distance dependencies and their importance in relation to question analysis are discussed in Section 7.2. In Section 7.3 I describe related work on recovering long distance dependencies from parser output. Section 7.4 presents a new method for recovering LDDs in parser output trees using reentrancies calculated on automatically generated and LDD-resolved f-structures. Section 7.5 describes the evaluation of this method and gives a comparison with some of the previous work. Section 7.6 describes some measures taken to improve the scores in the evaluation. Section 7.7 summarises and concludes.

Part of this work has been published in Judge et al. (2006).

7.2 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 (who, whom, etc.) or modifier (when, where, etc.) 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 7.1).

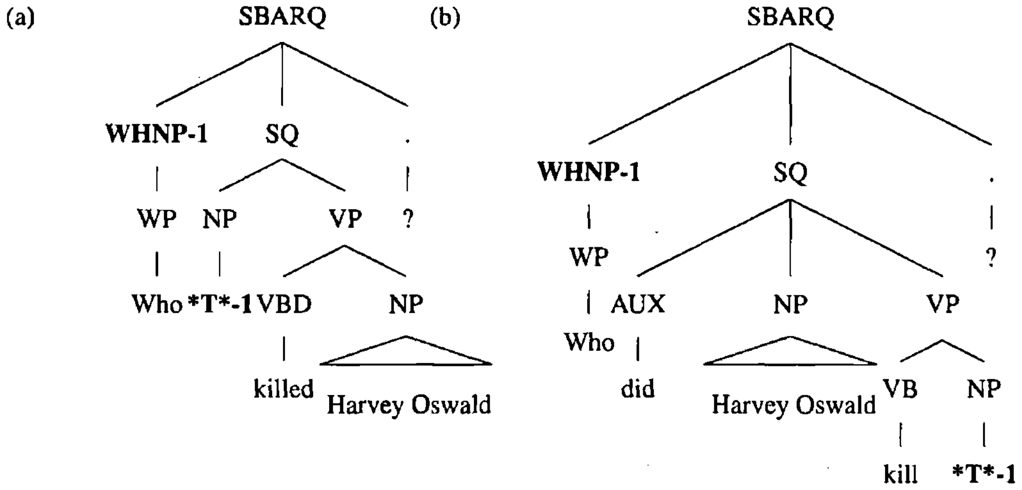


Figure 7.1: LDD resolved Penn-II/ATIS treebank style trees

With few exceptions[1] the trees produced by current treebank-based probabilistic parsers do not represent long distance dependencies but produce output without empty productions and coindexation such as in Figure 7.2.

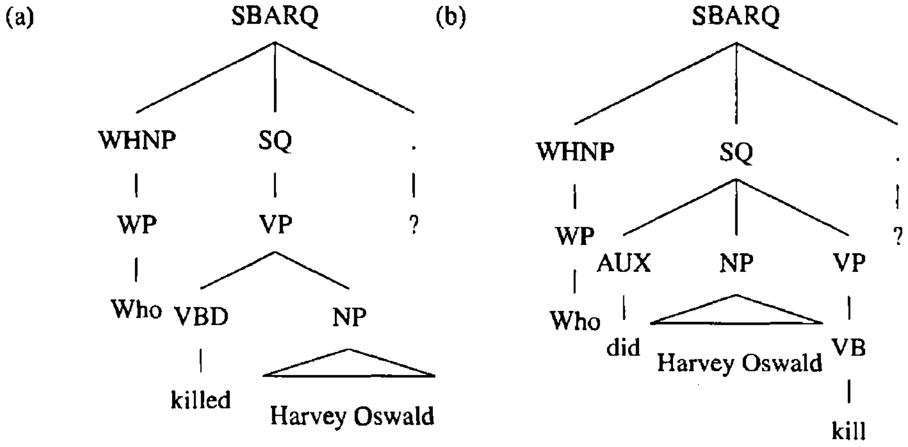


Figure 7.2: Parser output trees

[1] Collin's Model 3 computes a limited number of wh-dependencies in relative clause constructions.

For question analysis, long distance dependency information can be crucial in differentiating superficially similar questions. In order to provide a full analysis of questions, QuestionBank should also contain empty nodes and traces to represent LDDs. Adding this information in the form of traces and coindexed antecedents will also bring QuestionBank into line with established treebanks like the Penn-II Treebank and enable other Penn-II labelled data-based statistical and machine learning-based NLP approaches to use QuestionBank as a training and evaluation resource.

7.3 Related Work

Johnson (2002) presents a tree-based method for reconstructing long distance dependencies in Penn-II Treebank style parser output trees. His method uses tree patterns learned from the Penn-II Treebank to identify empty node insertion sites in the parser output. A tree pattern is the minimal tree fragment which connects a given empty node to its antecedent, and matching can only occur if a given tree is an extension of the pattern ignoring empty nodes. In evaluations on Section 23 of the Penn-II Treebank, Johnson's method resolves LDDs in parser output with a labelled bracketing precision and recall f-score of 68%.

Dienes and Dubey (2003) use a pre-processing method to identify where in the surface string an empty element is likely to occur. The input string is tagged using a "trace tagger" which adds gap information to the sentence. The sentence is then parsed with a modified parser which can thread this gap information through the parse tree to link the empty element and its antecedent. In experiments on Section 23 of the Penn-II Treebank this method resolves LDDs in parser output with a labelled bracketing precision and recall f-score of 74.6%. Dienes and Dubey (2003) also experiment with in-processing of empty nodes where the empty nodes are detected by the parser (which is an extension of Collins (1999) Model 3), however this method fails to surpass the accuracy of their pre-processing method.

Levy and Manning (2004) present a method to recover long distance dependencies from parser output. Their method uses feature classifiers in three phases to identify dislocated tree nodes and their origin site. This contrasts with Johnson's (2002) approach where empty node and antecedent recovery is treated as a single pattern matching task, and also with Dienes and Dubey (2003) who identify empty elements first, and the antecedents later in the parser. Evaluation on Section 23 of the Penn-II Treebank shows that the method is comparable to both Johnson (2002) and Dienes and Dubey (2003). Levy and Manning also assess cross-linguistic effectiveness of their approach, comparing English with a freer word order language, German. The results indicate that recovering LDDs is more difficult in German due to the freer word order.

A machine-learning approach to recover long distance dependencies arising out of WH-gaps is presented by Higgins (2003). This approach uses a subset of sentences extracted from the Penn-II Treebank which contain WH-gaps. From this data, for each sentence, subtree paths (similar to those used by Johnson (2002)) representing the path from the WH-phrase to the gap (empty element) were extracted. These paths were then used to train a HMM classifier which was applied to the test set. Higgins reports accuracy of 92% in experiments with his classifier on 1663 of the WH-gap sentences extracted from the Penn-II Treebank which were held out from the training data.

Campbell (2004) approaches LDD resolution using a rule-based method to reconstruct LDDs. His approach is based on the premise that the existence and location of empty categories is not determined by data observable in a corpus, but rather by linguistic principles. This is because empty categories (and their relation to possible antecedents) do not correspond to actual strings in the data, but rather they are part of the annotation put there by a linguist following linguistic annotation guidelines. This method, when tested on Section 23 of the Penn-II Treebank, resolves LDDs in parser output with a labelled precision and recall f-score of 76.7%.

Jijkoun and de Rijke (2004) present a dependency graph-based alternative to recovering long distance dependencies from parser output. In their method, the Penn-II Treebank trees are transformed into dependency graphs and a machine learning algorithm, TiMBL (Daelemans et al., 2003) is trained on the LDD resolved dependency graphs from the treebank trees. Parser output is then converted to dependency graphs and TiMBL adds the LDD information to the graphs generated from parser output. In tests on dependency graphs for Section 23 of the Penn-II Treebank this method has slightly higher precision than that of Dienes and Dubey (2003) but lower recall, resulting in an f-score of 74.6%.

7.4 Recovering Long Distance Dependencies in CFG Parser Output Using F-Structure Reentrancies

7.4.1 Motivation

QuestionBank was created with a view to offering it for use as a training and evaluation corpus for parser-based technology for question analysis. Experiments in Chapter 6 have shown that it is suitable for this purpose. However, in order to conform to the annotation guidelines (Bies et al., 1995) it needs to have trace information added to the trees so either an (semi-) automatic method of doing this, or manual annotation is necessary. The f-structure-based LDD resolution method of Cahill et al. (2004) can resolve LDDs in f-structures generated automatically from parser output very accurately, once the c-structure tree is of good quality. So in order to complete QuestionBank, and make it useful to other researchers in the area, I have developed a method to resolve LDDs in the hand-corrected QuestionBank trees which uses the f-structure based LDD resolution method of Cahill et al. (2004) to resolve the dependency, and then "reverse engineers" the syntactic component, based on information in the c-structure tree and the corresponding f-structure(s).

7.4.2 Method

Cahill et al. (2004) present a method for resolving LDDs at the level of Lexical-Functional Grammar f-structure without the need for empty productions and coindexation in parser output trees. Their method is based on learning finite approximations of functional uncertainty equations (regular expressions over paths in f-structure) and subcategorisation frames from an automatically f-structure annotated version of the Penn-II treebank and resolves LDDs at f-structure. The LDD resolution works on automatically generated f-structures output from an automatic f-structure annotation algorithm which is applied to LDD-unresolved c-structure parser output trees. The LDD module produces LDD resolved f-structures, with LDDs encoded by means of reentrancies in the f-structures.

In my work on restoring LDD information in the QuestionBank trees I use the f-structure based method of Cahill et al. (2004) to generate LDD resolved f-structures for (unresolved) QuestionBank trees then my system "reverse engineers" empty productions, traces and coindexation in the c-structure trees. I explain the process by way of a worked example.

I use the parser output trees in Figure 7.2 (without empty productions and coindexation), automatically annotate the trees 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 trees[2] and the LDD resolved f-structures in Figure 7.3.

[2] Lexical annotations are suppressed to aid readability.

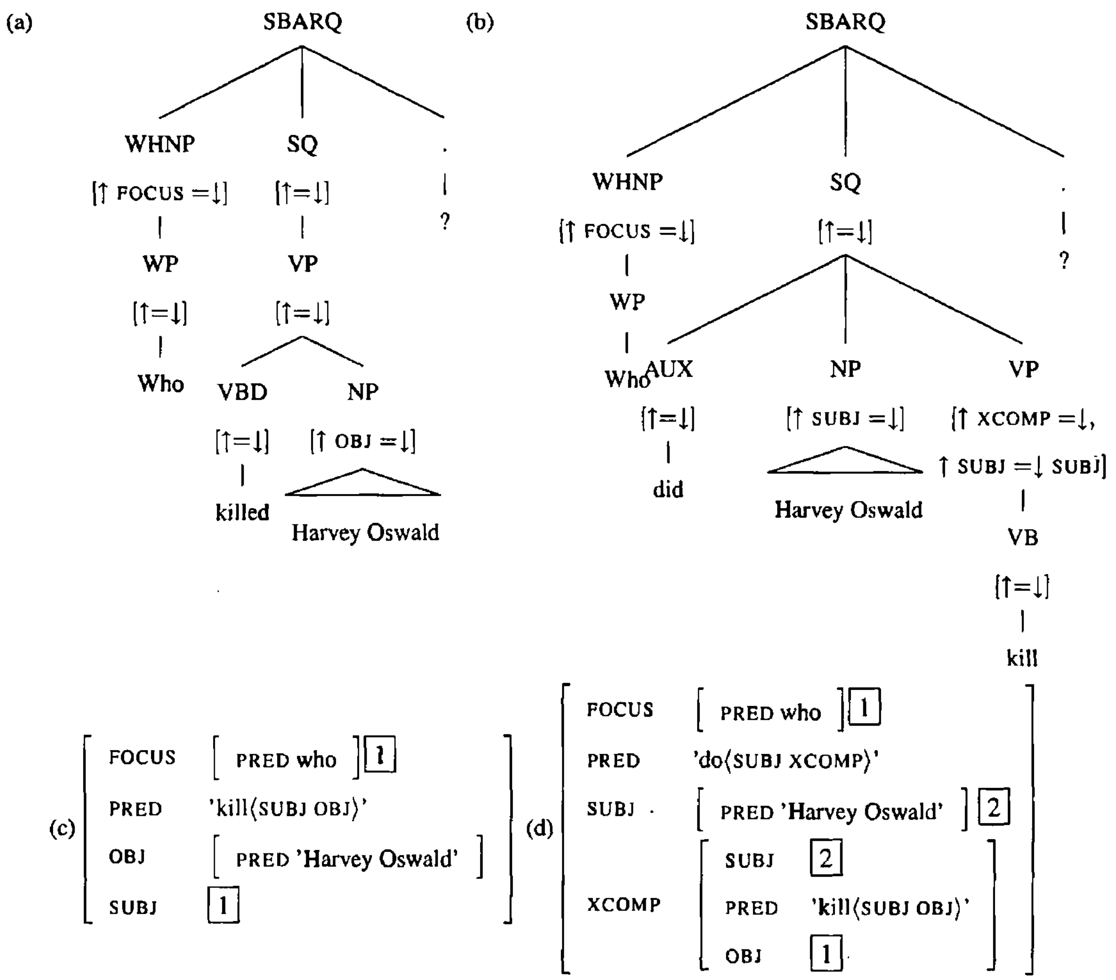


Figure 7.3: Annotated tree and f-structure

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 in Figure 7.3(c) and the FOCUS and OBJ function in 7.3(d). 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 in 7.3(c) is an argument of the PRED 'kill' and the verb form 'killed' in the tree. In order to reconstruct the corresponding empty subject XP node in the parser output tree 7.3(a), 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 CFG constituent corresponding to the governing PRED in the f-structure, in this case VP, indicated by ↑=↓ annotations in LFG. This establishes three anchor sites: VP, SQ and the top level SBARQ in 7.3(a).

The next step is guided by the annotated c-structure tree and uses an automatically f-structure annotated corpus as the basis for an APCFG (Section 2.3.3). From the automatically f-structure annotated Penn-II treebank (Sections 02-21), I extracted f-structure annotated PCFG rules expanding the three anchor sites whose RHSs contain exactly the information (daughter categories plus LFG annotations) already present in the tree in Figure 7.3(a) (in the same order) but which introduce an additional node (of any CFG category XP) annotated ↑ SUBJ =↓, located anywhere within the RHSs. Annotated APCFG rules are of the form

SBARQ[↑=↓]	→	WHNP[↑ FOCUS =↓] SQ[↑=↓] .	43
SBARQ[↑=↓]	→	WHNP[↑ FOCUS =↓] SINV[↑=↓] .	3
⋮			
SQ[↑=↓]	→	NP[↑ SUBJ =↓] VP[↑=↓]	40
SQ[↑=↓]	→	VBD[↑=↓] NP[↑ OBJ =↓] ADVP[↓ ELEM =↑ ADJUNCT]	1
⋮			
VP[↑=↓]	→	VBD[↑=↓] NP[↑ OBJ =↓]	3903
VP[↑=↓]	→	VBD[↑=↓] NP[↑ OBJ =↓] ADVP[↓ ELEM =↑ ADJUNCT]	56
⋮			

each with their associated frequencies (rightmost column in the above example). The LHSs of these rules correspond to the node(s) identified as potential anchor sites.

Among the appropriate rules,[3] I select the rule with the overall highest frequency and cut the rule into the tree in Figure 7.3 (a) at the appropriate anchor site (as determined by the rule LHS). In our case this selects SQ[↑=↓] → NP[↑ SUBJ =↓] VP[↑=↓] and the resulting tree is given in

[3] Not all of the rules shown are appropriate for the example used here. Only rules which introduce ↑ SUBJ = ↓ annotations are appropriate for the example, the other rules are given purely for illustrative purposes.

Figure 7.4. From this tree, it is now easy to compute that the tree node coindexed with the trace in Figure 7.4 is the WHNP annotated ↑ FOCUS =↓ following the reentrancy in the f-structure in Figure 7.3 (c).

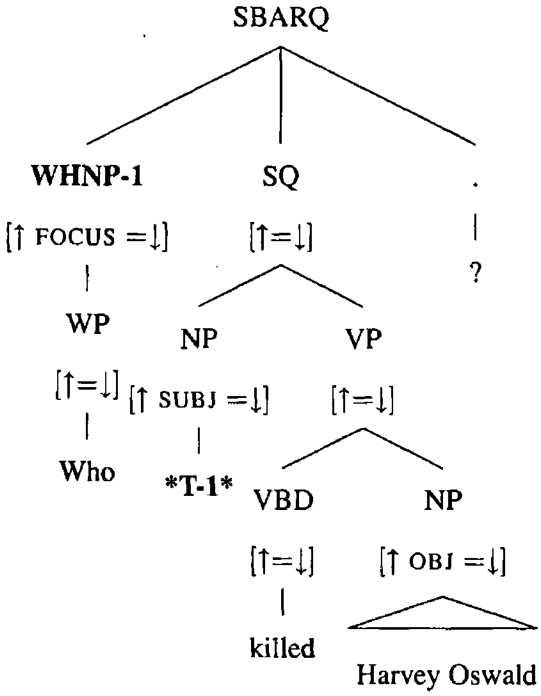


Figure 7.4: Resolved tree

Likewise the parser output in Figure 7.2(b) is annotated to give the tree and corresponding f-structure shown in Figure 7.5(a) and (b). The reentrancy between the FOCUS and OBJ of 'kill' results in an annotated CFG rule of the form

VP[↑=↓] → VB[↑=↓] NP[↑ OBJ =↓]

to be selected to insert an empty node labelled NP as the rightmost daughter of the VP. Using the reentrancy information in the f-structure, the empty node is coindexed with the WHNP producing the LDD resolved tree in Figure 7.5(c).

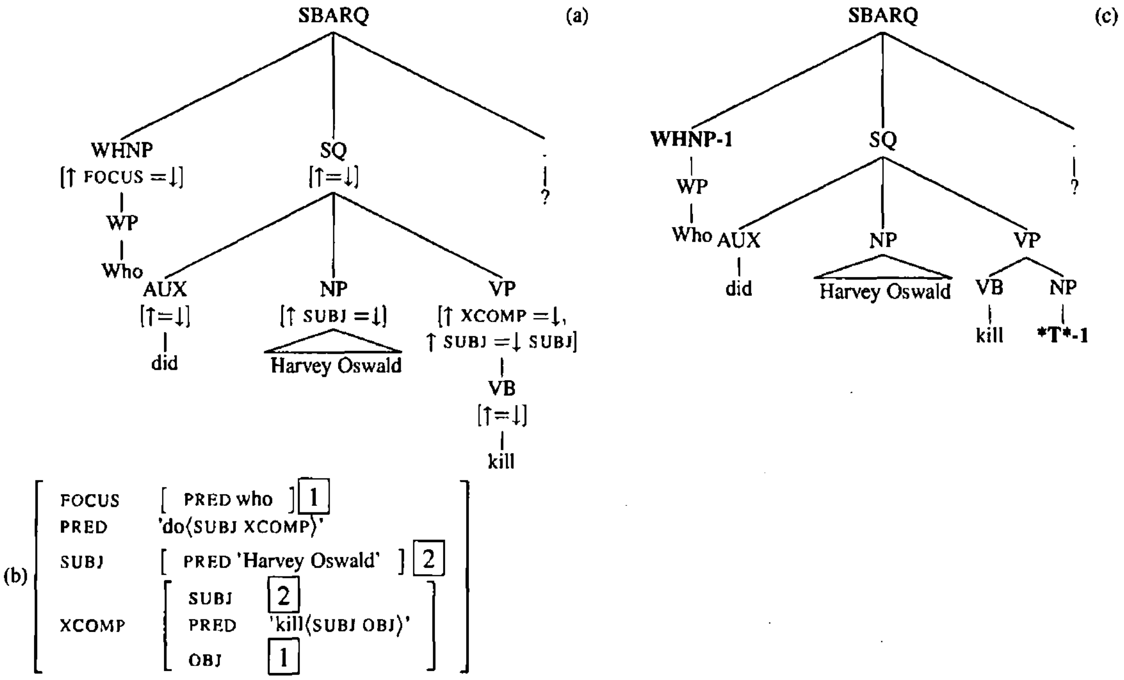


Figure 7.5: Another annotated tree and f-structure

7.5 Experiments and Evaluation

In order to evaluate this method of restoring traces, and to assess its suitability as a means to introduce trace information into QuestionBank, I carried out experiments using questions extracted from the ATIS corpus. There are a total of 213 questions in the ATIS corpus; however not all 213 questions are suitable for this evaluation. Many of the questions are syntactically analysed using X and FRAG constituents[4] and the automatic f-structure annotation algorithm of Cahill et al. (2004) will not be able to generate a proper f-structure for them. The ATIS tree in Figure 7.6 (a) and the corresponding parser output for the same string in Figure 7.6(b) illustrate this.

[4] The Penn-II Treebank guidelines give the following definitions for the X and FRAG labels: FRAG - Fragment. X - Unknown, uncertain, or unbracketable. X is often used for bracketing typos and in bracketing the...the-constructions.

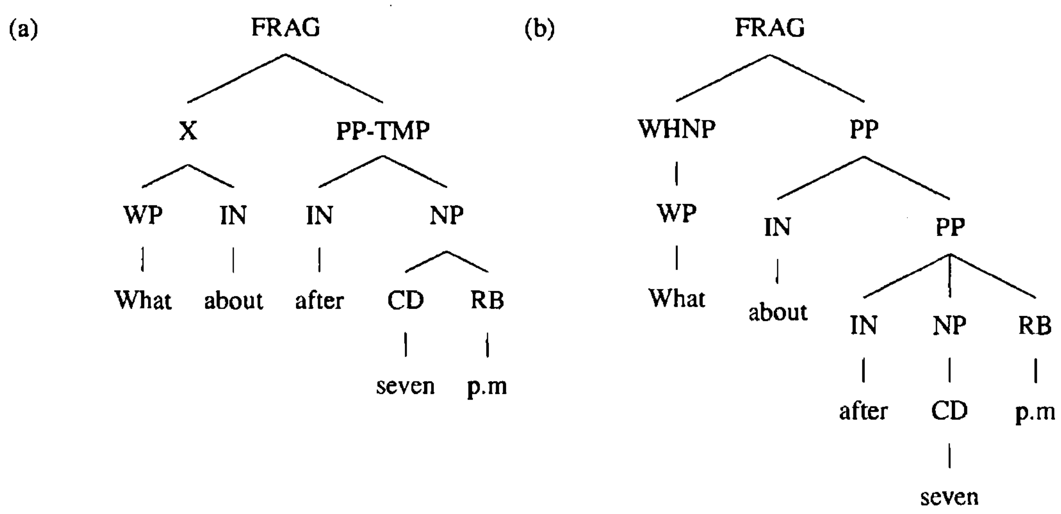


Figure 7.6: A Fragmented Question

Excluding trees containing X and FRAG constituents, I extracted 142 questions from the ATIS gold standard trees which encode a long distance dependency in the ATIS tree and submitted them to the automatic f-structure annotation algorithm of Cahill et al. (2004) to generate LDD resolved f-structures to be used as a gold standard against which to evaluate the LDD resolution algorithm described in Section 7.4.

In the first experiment, I delete empty nodes and coindexation from the ATIS gold standard trees and reconstruct them using my f-structure-based method on the preprocessed "perfect" ATIS treebank trees. In the second experiment, I parse the strings corresponding to the ATIS trees with Bikel's parser (trained on Sections 02-21 of the Penn-II Treebank and QuestionBank) and reconstruct the empty productions and coindexation on the parser output trees. In both cases I evaluate against the original (unreduced) ATIS trees (with empty productions and coindexation) and score if and only if all of insertion site, inserted CFG category and coindexation match. This is a stricter evaluation than in some of the previous work, and follows the evaluation method proposed in Campbell (2004).

This evaluation method has advantages over the string position-based method used by Johnson (2002) where the location of an inserted empty node is considered correct if the empty string which it governs occurs in the correct position in the surface string. Consider the sentence "When do you expect to finish?" with the bracketing shown below, and where 1 and 2 indicate possible locations for the trace of the WHADVP

$[_{WHADVP}$ When] do you $[_{VP}$ expect $[_{VP}$ to finish 1] 2]

1 structurally relates the queried temporal location to the finishing eventuality ("When do you finish"), whereas 2 relates it to the expecting eventuality ("When do you expect...?").

In a string position-based evaluation, if position 1 is the correct position for the trace according to the gold-standard, then, because both position 1 and 2 occupy the same string position, a system which inserts the trace in position 2 will not be penalised. The evaluation method I use considers both the label and parent category of inserted empty nodes and overcomes this shortcoming of string position-based evaluations.

	Gold Standard Trees	Parser Output
Precision	96.82	96.77
Recall	39.38	38.75
F-Score	55.99	55.34

Table 7.1: Scores for LDD recovery (empty nodes and antecedents)

Table 7.1 shows that the recall of the method is quite low at 39.38% while the accuracy is very high with precision at 96.82% on the original preprocessed 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%.

7.5.1 Comparison with Other Methods

The high precision of the method for recovering long distance dependencies in parser output using f-structure reentrancies is encouraging. The low recall, however, means that there is still room for much improvement. As I have noted before, questions are a very different text type compared to informative newspaper text like that found in the Penn-II Treebank. Part of the basis for my LDD recovery method is an APCFG extracted from an automatically f-structure annotated version of Sections 02-21 of the Penn-II Treebank, and it is possible that the recall underperformance on questions from the ATIS corpus is due to the difference in text type when dealing with questions.

To test this hypothesis I compared Johnson's (2002) and Higgins' (2003) systems with my own. I used the same 142 ATIS question test set and evaluate the two systems' performance on gold standard trees stripped of coindexation and functional tags and on parser output from Bikel's parser retrained on Penn-II Treebank WSJ Sections 02-21 and QuestionBank. Higgins' software used for these evaluations is a slightly improved version of the one used to generate the results published in the literature,[5] which I acquired through personal communication with the author. Johnson's software for the evaluations was downloaded from http://www.cog.brown.edu/~mj/Software.htm. The results are summarised in Table 7.2.

	Johnson 02			Higgins 03[6]			Judge 06		
	P	R	F	P	R	F	P	R	F
Stripped Gold Standard Trees	**97.14**	49.3	65.41	91.59	**69.01**	**78.71**	96.82	39.38	55.99
Parser Output	**96.97**	45.21	61.67	94.11	**67.61**	**78.69**	96.77	38.75	55.34

Table 7.2: LDD recovery (empty nodes and antecedents) compared

The results in Table 7.2 show that on the ATIS 142 question test set Johnson's method gives the highest precision and Higgins' method gives the highest recall. The f-structure reentrancy-based method used here performs almost as well as Johnson's (Penn-II Treebank-based) method

[5] Features related to the head of a constituent have been added. These allow the model to determine that something really is an extracted object if the VP it sees consists of a transitive verb with no overt object, rather than an intransitive verb with no object.

[6] Higgins' system does not output a syntactic category for empty nodes it produces. In these evaluations I assume that the inserted node is *always* of the right category. This is perhaps being overly generous to Higgins' system, however if I was to evaluate on the output from his system without this assumption the scores would all be zero, which I feel is unduly unfair towards the system.

in terms of precision, but is outperformed by both the other systems in terms of recall and f-score.

It is interesting to note here that Higgins' system, which focuses specifically on WH constructions, scores lower in precision on the question test set than the other LDD recovery systems which are designed for generic LDDs (including topicalisation phenomena etc.). However recall for Higgins' system is considerably higher than both of the others. Also interesting (and somewhat surprising) about the performance of Higgins' system is that there is a marked increase in precision, but a drop in recall when it is tested using parser output.

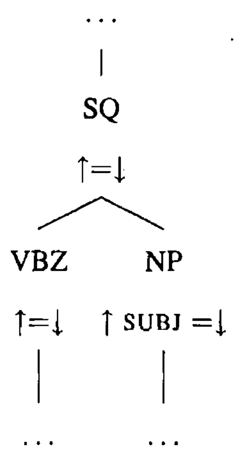
7.6 Improving LDD Recovery Accuracy

While the proposed method to recover long distance dependencies in parser output using f-structure reentrancies gives high precision on the ATIS question test set, the experiments in Section 7.5.1 show that the high precision and low recall performance is in fact common to both of the Penn-II Treebank based *general* LDD resolution methods (Johnson's and my own). Higgins' *WH specific* method outperforms the other methods in terms of recall, making up for the slightly lower precision scores and making it the best-performing system overall in terms of f-score.

In order to maximally exploit the use of a system to add the necessary long distance dependency information to QuestionBank (i.e. to minimise the need for manual correction and addition of LDDs), it would be preferable to have both high precision *and* recall scores. In order to achieve this, I used two complementary approaches to improve the LDD recovery system: I relaxed the constraints on the rule matching criteria slightly, and I added LDD resolved questions (these include LDD resolved trees correctly generated by the method described above, (a small number of) trees which were resolved incorrectly but manually corrected and trees which were not LDD resolved at all by the automatic method and hence were LDD resolved fully manually) in a bootstrapping approach to the corpus from which the APCFG rules are extracted. These approaches are described below.

7.6.1 Relaxing the Matching Criteria

The LDD recovery method described above uses very strict matching criteria for selecting rules from the APCFG to consider as candidates to use to insert an empty node. For example, say the algorithm is trying to restore a trace in a tree fragment (for a copular construction) like the one below



where the f-structure reentrancy indicates that the functional annotation $\uparrow$ XCOMP $=\downarrow$ is necessary and the APCFG contains the rule

SQ[$\uparrow=\downarrow$] $\rightarrow$ VBZ[$\uparrow=\downarrow$] NP[$\uparrow$ SUBJ $=\downarrow$] NP[$\uparrow$ XCOMP $=\downarrow, \uparrow$ SUBJ $=\downarrow$ SUBJ]

In this case the algorithm will not consider the rule above as a candidate to allow the insertion of an empty node labelled NP because the annotation on the second NP in the annotated rule ($\uparrow$ XCOMP $=\downarrow, \uparrow$ SUBJ $=\downarrow$ SUBJ) is not *exactly* the same as the annotation which the reentrancy is looking for ($\uparrow$ XCOMP $=\downarrow$). This is perhaps too strict in certain cases, so I modified the rule matching routine to allow a match when the annotation on candidate nodes subsumes the annotation sought. That is to say if the annotation required on a candidate empty node is x and a rule matching the subtree where an empty node is proposed introduces a node N with a set of annotations S, then N can be considered a candidate empty node for insertion if $x \in S$.

With the matching constraint relaxed in this way, the coverage of the algorithm increased considerably. However, the extra freedom in the matching criteria meant that a lot of extra candidates were being considered for insertion and very few of the nodes which were inserted were actually correct.

Instead of using the relaxed matching criteria for the annotation on nodes proposed for insertion, I chose to make the matching routine less strict than it was originally, but not as indiscriminate as in the previous attempt. I changed the matching routine to allow a node to match if its annotation is exactly the annotation sought, or, if there is a set of annotations on the node, the first annotation listed in the set of annotations matches the annotation needed. This rather crude heuristic had a surprisingly good effect on recall, which increased considerably, but also had a negative effect on precision, though not to as great an extent as the more relaxed criteria in the previous attempt. While this is not entirely satisfactory, further improvements can be made by adding question data to the APCFG, which I deal with next.

7.6.2 Tuning the APCFG to Questions

The LDD recovery method described in Section 7.4 is initially biased towards Penn-II Treebank style material as it relies on an annotated PCFG (APCFG) extracted from an f-structure annotated version of the Penn-II Treebank to determine what nodes to consider for insertion and where. I have shown in Chapter 4 that c-structure is more susceptible to domain/corpus dependence than f-structures, so it is not unexpected that a set of CFG rules (even though they may contain functional annotation) extracted from the Penn-II Treebank will underperform on questions.

If the annotated grammar contained more question specific constructions, then the question specific constructions would have higher frequencies and hence be chosen over other possible constructions by the node insertion routine. QuestionBank, as described in Chapter 5, contains a large quantity of question specific constructions, however, the trees do not contain any traces or long distance dependencies so initially they cannot be used to help improve precision and recall in the LDD recovery algorithm.

To resolve this issue, I used a process similar to that used in the creation of QuestionBank in the first place. I iteratively processed sections of QuestionBank with the LDD recovery algorithm, hand corrected and completed the output and after each iteration I annotated the corrected

LDD resolved trees with the automatic f-structure annotation algorithm of Cahill et al. (2004), added these annotated trees to the corpus from which the APCFG was extracted, and extracted a new APCFG for use in the next stage. In this way I was able to inductively add trace information to QuestionBank, and also increase the precision of the LDD recovery algorithm.

Algorithm 2 Adapt an Annotated PCFG for Resolving LDDs in Questions

```
repeat
    Resolve LDDs in a section of QuestionBank
    Manually correct and complete the automatically LDD resolved trees
    Automatically f-structure annotate manually corrected LDD resolved QuestionBank trees
    Add the annotated LDD resolved question trees to the corpus
    Extract a new APCFG
until All the data has been processed
```

Algorithm 2 shows the "bootstrapping" procedure used to induce trace information in QuestionBank. This was done in five stages and at each stage I evaluated the trace recovery program against a set of 100 randomly extracted hand corrected LDD resolved QuestionBank questions. The stages are outlined below

Baseline (0 QuestionBank Trees) Initially I took the trace restoration program described above and processed 400 QuestionBank trees. I hand corrected the 400 trees and randomly selected 100 of these 400 trees to use as an evaluation set.

Iteration 1 (300 QuestionBank Trees) I took the remaining 300 hand corrected LDD resolved trees, automatically annotated them with f-structure information using the automatic annotation algorithm of Cahill et al. (2004) and added the annotated trees to the corpus from which the APCFG used in trace recovery is extracted. I then extracted a new APCFG and processed the next 600 trees.

Iteration 2 (900 QuestionBank Trees) I corrected the 600 trees output by the system, annotated them, extracted a new APCFG and processed the next 1000 trees.

Iteration 3 (1900 QuestionBank Trees) I corrected the 1000 trees output by the system, annotated them, extracted a new APCFG and processed the next 1000 trees.

Iteration 4 (2900 QuestionBank Trees) I corrected the 1000 trees output by the system, annotated them, extracted a new APCFG and processed the next 1000 trees.

Iteration 5 (3900 QuestionBank Trees) I corrected the 1000 trees output by the system, annotated them, extracted a new APCFG.

7.6.3 Evaluating the Improved System

100 QuestionBank trees were randomly selected and held out as an LDD resolved question test-set to evaluate against during the LDD bootstrapping process. At each stage in the bootstrapping process I evaluated the LDD resolution given gold standard trees and parser output[7] against these hand corrected trees. The results for each stage are given in Table 7.3.

[7] Produced using the same parser/grammar combination as was used in Section 7.5 to generate parser output.

		Precision	Recall	F-Score
Baseline[8]	Gold Standard	45.07	71.00	55.13
	Parser Output	25.45	55.00	34.80
Iteration 1	Gold Standard	86.30	73.00	79.09
	Parser Output	82.46	57.00	67.41
Iteration 2	Gold Standard	86.30	73.00	79.09
	Parser Output	84.48	58.00	68.78
Iteration 3	Gold Standard	90.41	73.00	80.78
	Parser Output	83.05	59.00	68.99
Iteration 4	Gold Standard	90.41	73.00	80.78
	Parser Output	83.05	59.00	68.99
Iteration 5	Gold Standard	90.41	73.00	80.78
	Parser Output	83.05	59.00	68.99

Table 7.3: LDD resolution results for each stage of inducing LDDs in QuestionBank with relaxed matching constraint (Section 7.6.1)

The evaluations show that before any LDD resolved QuestionBank trees were added to the corpus from which the APCFG is extracted, the quality of LDD resolution in the questions is quite poor. The recall is greatly improved compared to the ATIS result in Table 7.1, but the precision is quite poor, and the f-score for the baseline evaluation is only 55.13% for gold standard QuestionBank trees stripped of traces and coindexation, and 34.80% for parser output. The evaluation after the first iteration of hand correction and reextraction of an extended APCFG shows a dramatic improvement on the baseline. In this case, 300 hand corrected LDD resolved QuestionBank trees have been added to the corpus from which the APCFG is extracted, and

[8] Note that the situation with respect to precision and recall of the system (high precision with low recall) has been reversed here in the baseline evaluation. This reflects the effect of relaxing the matching constraint (Section 7.6.1), the evaluations during subsequent iterations of the process reflect the result given both measures to improve performance (relaxing the matching constraint and including QuestionBank data).

already the results have improved considerably. Precision and recall are up in both evaluations. Recall, however, has improved to a much lesser extent than precision. The stripped gold standard QuestionBank tree input f-score increases to 79.09% and for the parser output it increases to 67.41%, an improvement of 23.96% over the initial stripped gold standard trees evaluation and 32.61% for the parser output evaluation.

The evaluations on the subsequent iterations show much less dramatic increases for both the stripped gold standard QuestionBank trees and parser output evaluations. The stripped gold standard trees evaluations hold steady at an f-score of 79.09% until the third iteration where it increases to 80.78% and stays constant throughout the remaining runs. The parser output evaluation increases in terms of recall until the third iteration where it remains constant at 59%, while precision spikes at 84.48% on the second iteration but then drops back and stabilises at 83.05% on the third and subsequent runs. The overall f-score for the parser output evaluations increases gradually until the third iteration (despite the precision spike and drop off) where it stabilises at 68.99%.

The dependencies in the 100 QuestionBank questions are all wh-movement dependencies. A breakdown of the results by the CFG category of the empty elements associated with the dependencies is shown in Table 7.4.

	Stripped Gold Standard			Parser Output		
	Precision	Recall	F-Score	Precision	Recall	F-score
NP	95.23	80.00	86.95	84.62	61.33	71.11
ADVP	0.00	0.00	0.00	0.00	0.00	0.00
ADJP	0.00	0.00	0.00	0.00	0.00	0.00

Table 7.4: Breakdown of LDD recovery results on 100 QuestionBank questions by CFG type

These results show that the LDD recovery performs best on recovering dependencies which introduce an empty element corresponding to an NP, achieving an f-score of 86.95%, and that

	Stripped Gold Standard			Parser Output		
	Precision	Recall	F-Score	Precision	Recall	F-score
Subj	96.43	87.09	91.52	100	35.48	52.38
Obj	84.62	76.74	80.49	81.58	72.09	76.54
Mod	0.00	0.00	0.00	0.00	0.00	0.00

Table 7.5: Breakdown of LDD recovery results on 100 QuestionBank questions by functional role of the empty element

while dependencies are recovered for other categories, those recovered in this evaluation are always incorrect. Manual inspection of these errors shows that while the position and coindexation of the inserted empty elements is correct, the wrong CFG category is assigned. Further investigation revealed that the cause of the empty element being assigned the wrong CFG category was, in fact, an incorrect f-structure analysis given by the automatic f-structure annotation algorithm of Cahill et al. (2004).

The recovered empty NP elements in Table 7.4 fulfill the role of subject or object in the questions. An alternative breakdown of the results by functional role is given in Table 7.5.

The results in Table 7.5 show that the precision for recovering subjects is considerably higher than for objects given both stripped gold standard input and parser output trees. However, recall is higher for objects than subjects given parser output trees. Interestingly, the f-score for recovering objects suffers less degradation from using parser output trees (f-score 76.54%) than recovering subjects (f-score 52.38%). The scores for modifiers are quite poor. These correspond to the ADVP and ADJP categories in Table 7.4 and so the low result can be explained by the inserted empty element being labelled with the wrong CFG category as a result of a bad f-structure analysis.

Upper Bound Estimation

The results for the updated system show a marked improvement on the earlier version of the system described in Section 7.4. To establish an upper bound and to answer the question whether the mistakes made by the system are caused by the LDD recovery algorithm or the automatic f-structure annotation which resolves the long distance dependencies in the first place I discount any of the questions in the gold standard where a manual inspection of the f-structures generated reveals that the automatic f-structure annotation algorithm has made a mistake, either in the f-structure or in the long distance dependency resolution (these amount to discounting a total 18 out of the 100 sentences). To eliminate the effect of parser error, I use stripped gold standard QuestionBank trees and automatically add empty nodes and their antecedents and evaluate against the gold standard. The results are shown in Table 7.6.

Precision	Recall	F-Score
98.57	85.70	91.69

Table 7.6: Upper bound results for f-structure reentrancy-based LDD resolution on 100 QuestionBank questions

In this evaluation precision increases to 98.57%, recall also increases (though to a lesser extent) to 85.70% giving an overall f-score of 91.69% for the LDD evaluation system using only those sentences where a properly resolved f-structure is generated by the automatic f-structure annotation algorithm of Cahill et al. (2004).

7.6.4 ATIS Evaluation

The more relaxed matching constraint and the added question data in the APCFG have greatly improved the overall performance of the LDD recovery algorithm. To find out how much these measures have contributed to improving the result, I revisited the earlier evaluations against the 142 ATIS sentence gold standard. Again I test the performance on both gold standard ATIS trees stripped of empty nodes and coindexation and parser output. The results of these evaluations, along with the scores for the original system, as well as Johnson's and Higgins' systems are given in Table 7.7.

	Stripped Gold Standard			Parser Output		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Johnson (2002)	**97.14**	49.30	65.41	**96.97**	45.21	61.67
Higgins (2003)	91.59	69.01	78.71	94.11	67.61	78.69
Judge (2006) basic	96.82	39.38	55.99	96.77	38.75	55.34
Judge (2006) best	89.29	**79.43**	**84.07**	89.29	**79.43**	**84.07**

Table 7.7: LDD recovery results for the improved LDD recovery algorithm on 142 ATIS question test set

Table 7.7 compares the performance of the improved LDD resolution system on the 142 ATIS question test set with the results in Table 7.2. The results show not only a substantial improvement for my system, but also that the improved system using QuestionBank material is the best performing of the four LDD recovery systems on the ATIS question test set. Interestingly, the results for the improved system on both stripped gold standard trees and parser output are the same. This is surprising. It is possibly due to similarities between some of the questions in ATIS, which differ only in terms of a place name or day, e.g. "What flights are there from X to Y?", "What flights leave X on DAY Y?". Closer examination of the parser output reveals that the parser output in this case is of very high quality and is structurally very similar to the gold standard trees. The differences between the parser output and gold standard trees are in constituents which do not affect my c-structure LDD insertion method and the correct functional analysis is still assigned and correct f-structure reentrancies representing LDDs are produced for the f-structure derived from the parser output.

The ATIS evaluation is possibly easier than the evaluation against the 100 QuestionBank questions as is indicated by the higher score achieved by each system given both gold standard input and parser output for the ATIS evaluation when compared with the results for testing against the 100 QuestionBank questions (Table 7.8). The precision for the improved system has dropped by around 7.5% on both evaluations, however the substantial gains in recall (just over 40% in each evaluation) make it the best performing system overall in terms of f-score. Johnson's system, however, still has the highest precision of all four systems.

For a final comparative evaluation of the systems, I tested Johnson's and Higgins' systems on the QuestionBank 100 question test set I used to monitor progress while bootstrapping trace information in QuestionBank. Table 7.8 compares these two systems with my improved system on these 100 questions.

	Stripped Gold Standard			Parser Output[9]		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Johnson (2002)	76.19	42.00	54.15	6.00	16.00	8.73
Higgins (2003)	**94.52**	**73.00**	**82.38**	79.03	**62.00**	**69.49**
Judge (2006) best	90.41	**73.00**	80.78	**83.05**	59.00	68.99

Table 7.8: Comparison of 3 LDD recovery methods on 100 QuestionBank trees

The results in this evaluation clearly show that the systems with question specific tuning, my improved system and Higgins' wh-specific system, are performing better on the question data than Johnson's Penn-II Treebank-based system.[10] Johnson's system performs particularly badly on parser output in this evaluation with both precision and recall very low. Higgins' system has the best overall performance in both evaluations, with my system scoring similarly.

The results in Table 7.8 show that, on the 100 question testset taken from QuestionBank, Higgins' system performs better than my f-structure reentrancy-based method of recovering

[9] Johnson's system did not output any coindexation on the traces that were inserted so the precision and f-score for his system here on parser output should be zero. However since I have been lenient towards Higgins' system with regards to labelling traces, I have included results for Johnson's system assuming that coindexation is correct when traces are inserted in the correct position in the tree.

[10] Johnson's software is retrainable. However, the code to extract new trace patterns for his software would not compile on a number of systems I attempted to do so on. Attempts to contact the author to resolve this issue received no response, so unfortunately I am unable to provide results for Johnson's software retrained using LDD resolved questions.

LDDs on both gold standard tree input stripped of traces and coindexation and parser output trees. These scores are, however, calculated giving the benefit of the doubt to Higgins' system, because it does not assign a CFG category to the empty nodes it inserts, and for the purposes of the evaluation I have assumed that the correct category is *always* assigned. This biases the evaluation towards Higgins' system, so I performed a further evaluation on the 100 QuestionBank questions with my own system where I do not penalise where the CFG category assigned to an empty node is incorrect. The results are given in Table 7.9.

	Stripped Gold Standard			Parser Output		
	Precision	Recall	F-Score	Precision	Recall	F-score
Higgins (2003)	94.52	**73.00**	82.38	79.03	**62.00**	69.49
Judge (2006) best	**100**	**73.00**	**84.39**	**100**	59.00	**74.21**

Table 7.9: Comparison of results for LDD resolution on 100 QuestionBank questions for my system and Higgins' system with the CFG category constraint relaxed for *both* systems

The results in Table 7.9 show that if I relax the evaluation constraints for my own system as well as Higgins', the the precision for my system in this evaluation increases to 100% resulting in f-scores greater than those of Higgins'.

7.7 Conclusion

In this chapter I have introduced a new method for recovering long distance dependency information (traces and coindexation) in parser output. This method uses the automatic f-structure annotation algorithm of Cahill et al. (2004) to generate an LDD resolved f-structure for the parser output c-structure tree, and, using an f-structure annotated PCFG (APCFG) to guide the process, "reverse engineers" the corresponding empty node and coindexation on the antecedent in the c-structure tree. This method proved to be highly accurate (96.77% precision), but to have low recall (38.75%) in initial tests on questions extracted from the ATIS corpus.

A comparison of my method with others showed that this high precision low recall result was typical of Penn-II Treebank-based systems in this evaluation. My system and Johnson (2002) performed similarly (though Johnson's results were better) with Higgins (2003), which focuses on wh- constructions in the Penn-II Treebank, outperforming both other systems with f-scores of 78.71% on gold standard ATIS trees stripped of empty nodes and coindexation, and 78.69% on parser output for the same sentences.

To improve the performance of my LDD recovery system I experimented with relaxing the constraints on considering nodes for insertion. This improved recall, but had a negative effect on precision. To remedy this, I added question data to the corpus from which the APCFG is derived. This question data was created inductively using QuestionBank as a source of unresolved questions which were processed using the LDD recovery system and a similar process-correct-retrain bootstrapping procedure used to create QuestionBank from raw data. This had two positive outcomes: first, it added traces and coindexation to the QuestionBank question trees, and second, it tuned the APCFG used in my LDD recovery system to questions. A series of evaluations carried out during the induction process show that the largest increase in accuracy occurred when the first section of hand-corrected LDD resolved questions was added to the APCFG training corpus. Subsequent additional data had a positive impact, but not to the same degree.

The combination of the change in matching criteria and the APCFG containing question data dramatically improved performance of my LDD recovery system. The precision and recall in evaluations on the ATIS test set that the earlier version was evaluated on increased giving an f-score on both stripped gold standard trees and parser output of 84.07%. This improved result means that my system outperforms Higgins' system in this evaluation by 5.36% on stripped gold standard trees and 5.38% on parser output.

An evaluation of all three LDD recovery systems on a QuestionBank-based test set shows that my (improved) system and Higgins' system perform very well on the question data, with high precision and recall for both stripped gold standard trees and parser output input, but Johnson's system performs particularly poorly with an f-score of 54.15% given stripped gold standard trees and 8.73% given parser output. Higgins' system scores highest in this evaluation, this is, however, under the assumption that Higgins' system *always* labels the inserted trace correctly. When I make the same assumption with respect to my own system and reevaluate, the results show that, my system outperforms Higgins' with an f-score of 84.39% on stripped gold standard input and 74.21% on parser output. These results, and those for the ATIS-based evaluations in Table 7.7 suggest that for the question-based evaluations conducted here, my f-structure reentrancy-based method of LDD recovery performs better.

The comparison of my improved system with Johnson's and Higgins' systems is biased, as I was unable to retrain neither Johnson's nor Higgins' system with QuestionBank data. This means that unlike my system they were unable to benefit from the QuestionBank data. The expectation would be that their results would improve and indeed QuestionBank is designed to provide a resource for retraining systems such as Johnson's and Higgins'. I hope to be able to explore this in future work.

Chapter 8

Conclusion and Future Work

Modern information retrieval systems are starting to employ linguistic analysis to a greater degree. Question answering systems, as a refinement of standard document IR, use more in-depth representations and sophisticated linguistic analysis techniques to retrieve precise answers to a direct question. Linguistic analysis captures information contained in natural language strings that allows these systems to pinpoint precise information in a text. Linguistic analysis can differentiate subtle (but important) differences between sentences, for example the subject/object (agent/patient) distinction in the focus of the questions "Who killed Harvey Oswald?" and "Who did Harvey Oswald kill?" and in declarative sentences containing appropriate answers to the questions "Jack Ruby killed Harvey Oswald." and "Harvey Oswald killed JFK."[1] However, a lack of linguistic resources for research focusing on questions and a lack of questions in standard training corpora and test sets means that some state-of-the-art analysis tools such as parsers underperform considerably on question material.

In this thesis I have presented work to assess and improve the state-of-the-art regarding the accurate analysis of questions into two levels of linguistic representation, CFG parse trees and LFG f-structures, as well as working towards addressing the lack of resources for question-focused analysis through the creation of QuestionBank, a parse-annotated corpus of questions

[1] The truth of this statement is the subject of much debate. I use the example for illustrative purposes only.

for use in the development of parser-based linguistic tools for question analysis. The research in this thesis has:

- shown that for current state-of-the-art Penn-II trained probabilistic parsers analysing questions is an instance of domain variation. This domain variation is more extreme than that observed in previous work on parser domain variance, however it can be treated in the same way.
- shown that the domain variance observed affects both c- and f-structure analyses but not to the same degree.
- established that the automatic f-structure annotation algorithm of Cahill et al. (2004) is complete with respect to domain variation experiments on data from the ATIS corpus.
- developed a training and evaluation corpus for developing question-focused parser-based linguistic resources, QuestionBank.
- compared the baseline parsing performance of a state-of-the-art parser trained on Sections 02-21 of the Penn-II Treebank tested on both Section 23 of the Penn-II Treebank and a question test set from QuestionBank with that of the same parser trained on the original training set plus QuestionBank.
- investigated the effects of varying the amount of Penn-II Treebank and QuestionBank data in the parser's training set when testing on both QuestionBank and Penn-II Treebank-based test sets.
- developed a method for restoring Long Distance Dependency trace and coindexation information in parser output c-structure trees using LFG f-structure reentrancies.
- compared this method of restoring LDD trace information in questions from QuestionBank with two existing methods (Johnson, 2002; Higgins, 2003).

- successfully used the LDD restoration method developed to inductively restore LDD trace information in the hand corrected parser output trees in QuestionBank.

The main resource established in this thesis is QuestionBank and the main result is that QuestionBank can be used in conjunction with the Penn-II Treebank to train a parser that can parse both questions and informative text with state-of-the-art accuracy. Below I summarise the findings and results of this research.

The Penn-II Treebank (Marcus et al., 1994), due to its low proportion of questions, is unsuitable for training a parser to analyse questions. The ATIS corpus (Hemphill et al., 1990) contains a much higher proportion of questions and is suitable for use in domain variation experiments testing the parser's performance on questions. In experiments with three state-of-the-art parsers (Collins, 1999; Charniak, 2000; Bikel, 2002) trained on Sections 02-21 of the Penn-II treebank, I have shown that the domain variance presented by the ATIS corpus is more severe than the variance that was observed in previous work by Gildea (2001) on parsing Brown Corpus (Kucera and Nelson, 1967) data, with parser performance for parsing the whole ATIS corpus dropping to a labelled precision and recall f-score of 72.45% for Collins' Model 2, 63.64% for Charniak and 69.94% for Bikel, an average drop of almost 20% when compared to the same parsers tested on Section 23 of the Penn-II Treebank. A small portion of this drop in performance for Charniak's parser (4.48%) can be attributed to the lack of punctuation in the ATIS corpus, however this is not true for Collins' and Bikel's parsers. As was observed in Gildea's work, the parsers' performance can be boosted on the out of domain test material by adding domain appropriate material to the parsers' training set. Retraining experiments on Charniak's and Bikel's parsers show significant gains on the baseline evaluations, giving best run labeled precision and recall f-score results on an ATIS-based test set of 84.69% for Charniak and 85.65% for Bikel. This is surprising given the relative difference in size between the ATIS corpus and the Penn-II Treebank, and it suggests that a question training corpus of similar size to the Penn-II Treebank may not be needed to produce good parsing results for questions.

I have examined the effect of domain variance on automatic c- and LFG f-structure analyses. Parsing and automatically annotating ATIS sentences with f-structures using the Penn-II trained "off the shelf" version of Bikel's parser and the automatic f-structure annotation algorithm of Cahill et al. (2004) results in a labelled bracketing c-structure f-score of 70.25% when the c-structure trees are evaluated against 100 ATIS gold standard trees and a preds-only dependency f-score of 62.95% when evaluated against hand-crafted f-structures for the 100 ATIS sentences. In this instance of domain variation the annotation of question specific relations like FOCUS and TOPICREL is particularly poor. In order to resolve the underperformance issue of the pipeline parsing and f-structure annotation algorithm architecture only the parser needed to be retrained. Retraining the parser on appropriate data from the ATIS corpus improves the c-structure f-score by 12.89% and the preds-only dependency f-score by 13.82%, and improves the quality of FOCUS and TOPICREL annotations dramatically. Interestingly, the retrained system, when back-tested on data in the original domain (DCU 105) shows no negative effects in both c- and f-structure evaluations as a result of the retraining. This suggests that the automatic f-structure annotation algorithm of Cahill et al. (2004) is complete with respect to the domain variation observed here and supports the view that f-structures are a more abstract representation of the information contained in a sentence, which is less affected by domain variation.

The significant improvements on out of domain c- and f-structure analysis as a result of parser retraining are encouraging given the size of the ATIS "question" corpus used in the experiments. Due to its size, and composition, however, the ATIS corpus is not entirely representative of questions. To address this I have created a corpus of parse-annotated questions, QuestionBank, from the TREC QA evaluation test sets and a question classifier training set provided by the Cognitive Computation Group at the University of Illinois Urbana-Champaign. QuestionBank was rapidly induced from raw data using a parse-correct-retrain bootstrapping method to simultaneously induce both a question treebank and a better question parser. This process generated 4000 Penn-II Treebank style trees in 3 months.

Experiments with QuestionBank show that a parser trained only on QuestionBank can parse questions accurately, with an f-score of 88.82% in a 10-fold cross-validation on QuestionBank, but performs badly on informative text in Section 23 of the Penn-II Treebank, with an f-score of only 59.79%. If QuestionBank is used in conjunction with the Penn-II Treebank as a training resource, the parser performance on Section 23 is comparable to when trained only on the Penn-II Treebank data (82.39%) and gives a 0.93% improvement on the grammar trained only on QuestionBank when parsing questions. This slight improvement shows that question parsing can gain something (however small) from extracting information from a corpus of informative text like the Penn-II Treebank. Using ablation experiments I have shown that the amount of data in QuestionBank does not constitute an absolute upper bound for question parsing, but that it is sufficiently large a resource that the potential gain from enlarging the question corpus does not justify the effort involved. From this I conclude that QuestionBank is sufficiently large for the task which it was intended for: to provide a training and evaluation corpus for parser-based linguistic analysis of questions.

I have developed a new method to restore Long Distance Dependency information in parser output c-structure trees using f-structure reentrancies assigned automatically to the parser output by the automatic f-structure annotation algorithm of Cahill et al. (2004). This process outperforms two existing systems for the same task, one syntax-based, and one using machine learning, in evaluations on sample questions extracted from the ATIS corpus, with an f-score of 84.07% on both parser output or gold standard trees stripped of trace information and also on QuestionBank questions, with an f-score of 80.78% on stripped gold standard trees and 68.99% given parser output.

The process of developing the LDD restoration algorithm to perform so well on the questions also helped the development of QuestionBank. The hand-corrected parser output trees in QuestionBank contained no long distance dependency information as the parser (Bikel, 2002) does not output this information in the trees. I used process-correct-retrain bootstrapping passes over sections of QuestionBank to restore trace information in the QuestionBank trees, and also improve the quality of the trace restoration algorithm.

8.1 Future Work

The main goal of this research has been to improve the quality of automatic linguistic analysis of questions and to provide the resources for other researchers to easily do so as well. In a larger context this work fits into ongoing work at the NCLT on acquiring wide-coverage, deep, constraint-based LFG grammars from treebanks, showing that the systems developed for English are robust and easily adaptable to a new domain.

The research in this thesis has looked at the underperformance of automatic linguistic analysis tools on questions as an instance of domain variation. Domain variation presents a wide range of possibilities for further work. The automatic f-structure annotation algorithm of Cahill et al. (2004) is complete with respect to the domain variation studied in Chapter 4. However, since the work presented here is the first research on domain variance for the automatic f-structure annotation algorithm, it is not necessarily the case that this will hold true for other domains of application.

Nothing in the methods used here precludes the possibility of application to another language, or to the adaption of a resource for one language to that of another linguistically similar language. For example one might consider a parser for Spanish parsing Portuguese as an instance of (truly "extreme") domain variation. It is possible that, using the bootstrapping and retraining method used in this research, a training corpus for Portuguese could be induced from raw text using a parser for Spanish.

To date, I have not been able to integrate the improvements in c- and f-structure analysis shown here with a working QA system to evaluate the effects on a working system. There are many ways in which this could be done depending on the system and how exactly it uses linguistic analysis. In a less involved system the improvements to c-and f-structure analysis could be used for tasks like disambiguation and query reformulation. In a more linguistically involved QA system, the improved parser performance can help the extraction of logical representations (possibly in the form of f-structures) for answer identification and justification.

Exploring this avenue is something I would like to pursue in the future, perhaps through the development of a linguistically motivated QA system, which relies on linguistic information to retrieve documents and answers instead of the current favoured strategy which relies on a boolean keyword matching retrieval engine for document retrieval. Developing such a system opens up a number of new avenues for exploration, for example:

- Tree or f-structure-based indexing and retrieval of documents,
- Tree or f-structure-based answer identification and ranking,
- Question reformulation based on syntactic/functional information as a fallback measure.

all of which rely heavily on accurate linguistic analysis of both questions and informative text into linguistic representations such as CFG trees or f-structures.

In its current form QuestionBank only provides a syntactic analysis for questions. Previous work (Burke, 2006) and experiments in Chapter 4 have shown that the automatic f-structure annotation algorithm of Cahill et al. (2004) has a high upper bound when given LDD resolved treebank trees as input. Based on this premise, an f-structure version of QuestionBank could be generated automatically from the parse-annotated version currently available. This would provide an automatically generated training/evaluation resource for question focused LFG-based resources.

Hockenmaier (2003a) presents an algorithm to transform phrase structure trees from the Penn Treebank to CCG derivations, creating a CCG version of the Penn Treebank, CCGBANK. Since QuestionBank follows the same annotation principles as the Penn-II Treebank (Bies et al., 1995) it should be possible to create a version of QuestionBank consisting of CCG derivation trees instead of phrase structure trees. Such a resource could supplement the CCG "What. . .?" question corpus already developed (Clark et al., 2004), making it more representative of different types of questions.

Before either the f-structure or CCG derivation tree versions of QuestionBank could be considered, the current version needs to be checked for errors. While every effort was made to ensure the accuracy and consistency of the annotation of the questions in QuestionBank the time constraints on the project did not allow for the corpus to be checked and validated by additional annotators. As such QuestionBank is currently only in a beta release stage. Before a final version can be made available both the trees and the long distance dependencies need to be verified by *at least* one additional annotator.

Bibliography

Bies, A., Ferguson, M., Katz, K., and MacIntyre, R. (1995). Bracketing Guidelines for Treebank II Style Penn Treebank Project. Technical Report, University of Pennsylvania, Philadelphia, PA.

Bikel, D. M. (2002). Design of a Multi-Lingual, Parallel-Processing Statistical Parsing Engine. In *Proceedings of Human Language Technology (HLT) 2002*, pages 24–27, San Diego, CA.

Bikel, D. M. (2004). *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. PhD thesis, University of Pennsylvania, Philadelphia, PA.

Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingira, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorinim, B., and Strzalkowski, T. (1991). A Procedure for Quantatively Comparing the Coverage of English Grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*, pages 306–311, Pacific Grove, CA.

Black, E., Jelinek, F., Lafferty, J. D., Magerman, D. M., Mercer, R. L., and Roukos, S. (1993). Towards History-Based Grammars: Using Richer Models for Probabilistic Parsing. In *Meeting of the Association for Computational Linguistics*, pages 31–37, Columbus, OH.

Bresnan, J. (2001). *Lexical-Functional Syntax*. Blackwell, Oxford.

Brin, S. and Page, L. (1998). The Anatomy of a Large-Scale Hypertextual (Web) Search Engine. *Computer Networks and ISDN Systems*, **30**(1–7):107–117.

Burke, M. (2006). *Automatic Annotation of the Penn-II Treebank with F-Structure Information.* PhD thesis, School of Computing, Dublin City University, Dublin, Ireland.

Burke, M., Cahill, A., O'Donovan, R., van Genabith, J., and Way, A. (2004). The Evaluation of an Automatic Annotation Algorithm against the PARC 700 Dependency Bank . In *Proceedings of the Ninth International Conference on LFG*, pages 101–121, Christchurch, New Zealand.

Burke, R. D., Hammond, K. J., Kulyukin, V. A., Lytinen, S. L., Tomuro, N., and Schoenberg, S. (1997). Question Answering from Frequently Asked Question Files: Experiences with the FAQ Finder System. *AI Magazine*, **18**(2):57–66.

Cahill, A. (2004). *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations.* PhD thesis, School of Computing, Dublin City University, Dublin, Ireland.

Cahill, A., Burke, M., O'Donovan, R., van Genabith, J., and Way, A. (2004). Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 320–327, Barcelona, Spain.

Cahill, A., McCarthy, M., van Genabith, J., and Way, A. (2002a). Automatic Annotation of the Penn-Treebank with LFG F-Structure Information. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC02) workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*, pages 8–15, Las Palmas, Canary Islands, Spain.

Cahill, A., McCarthy, M., van Genabith, J., and Way, A. (2002b). Parsing with PCFGs and Automatic F-Structure Annotation. In Butt, M. and King, T. H., editors, *Proceedings of the Seventh International Conference on LFG*, pages 76–95, Stanford, CA. CSLI Publications.

Campbell, R. (2004). Using Linguistic Principles to Recover Empty Categories. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL04)*, pages 645–652, Barcelona, Spain.

Charniak, E. (1997). Statistical Parsing with a Context-Free Grammar and Word Statistics. In *AAAI/IAAI*, pages 598–603, Providence, RI.

Charniak, E. (2000). A Maximum Entropy Inspired Parser. In *Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2000)*, pages 132–139, Seattle, WA.

Chomsky, N. (1973). Conditions on transformations. In Anderson, S. R. and Kiparsky, P., editors, *A festschrift for Morris Halle*, New York. Holt, Rinehart & Winston.

Civit, M. and Martí, M. A. (2004). Building Cast3LB: A Spanish treebank. *Research on Language and Computation*, 2(4):549–574.

Clark, S. and Curran, J. R. (2004). Parsing the WSJ Using CCG and Log-Linear Models. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL04)*, pages 103–110, Barcelona, Spain.

Clark, S., Steedman, M., and Curran, J. R. (2004). Object-Extraction and Question-Parsing using CCG . In Lin, D. and Wu, D., editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 111–118, Barcelona, Spain.

Collins, M. (1996). A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 184–191, Santa Cruz, CA.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA.

Crouch, R., Kaplan, R. T., King, T., and Riezler, S. (2002). A comparison of evaluation metrics for a broad coverage parser . In *Beyond PARSEVAL Workshop, Language Resources and Evaluation (LREC)*, pages 67–74, Las Palmas, Canary Islands, Spain.

Daelemans, W., Zavrel, J., van der Sloot, K., and van den Bosch, A. (2003). TiMBL: Tilburg Memory Based Learner, version 5.0, Reference guide. Technical Report 03-10, ILK, Tilburg University, The Netherlands.

Dalrymple, M. (2001). *Lexical-Functional Grammar.* San Diego, CA; London. Academic Press.

Dienes, P. and Dubey, A. (2003). Antecedent Recovery: Experiments with Trace Tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP03)*, pages 33 – 40, Sapporo, Japan.

Emms, M. (2005a). Adapting Tree Distance to Answer Retrieval and Parser Evaluation. In *Proceedings of the Workshop on Intelligent Linguistic Technologies (ILINTEC) at the 2005 International MultiConference in Computer Science and Computer Engineering (IMCSE 05)*, pages 53–59, Las Vegas, NV.

Emms, M. (2005b). Tree Distance in Answer Retrieval and Parser Evaluation. In Sharp, B., editor, *Proceedings of The Second International Workshop on Natural Language Understanding and Cognitive Science (NLUCS 05)*, pages 155–160, Miami, FL.

Frank, A. (2000). Automatic F-Structure Annotation of Treebank Trees. In Butt, M. and King, T. H., editors, *Proceedings of the Fifth International Conference on LFG*, pages 140–160, Stanford, CA. CSLI Publications.

Gildea, D. (2001). Corpus Variation and Parser Performance. In Lee, L. and Harman, D., editors, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 167–202, Pittsburgh, PA.

Green, B. F., Wolf, A. K., Chomsky, C., and Laughery, K. (1961). BASEBALL: An Automatic Question Answerer. In *Proceedings of the Western Joint Computer Conference*, pages 219–224. Reprinted in Grosz et al. (1986) pages 545–549, Los Angeles, CA.

Grosz, B., Jones, K. S., and Webber, B., editors (1986). *Readings in Natural Language Processing*. Morgan Kaufmann, California.

Harabagiu, S. M., Moldovan, D. I., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R. C., Girju, R., Rus, V., and Morarescu, P. (2000a). FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, pages 479–489, Gaithersburg, MD.

Harabagiu, S. M., Pasca, M., and Maiorano, S. J. (2000b). Experiments with Open-Domain Textual Question Answering. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 292–298, Saarbrücken, Germany.

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

Higgins, D. (2003). A Machine Learning Approach to the Identification of WH Gaps. In *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL03)*, pages 99–102, Budapest,Hungary.

Hirschman, L. and Gaizauskas, R. (2001). Natural Language Question Answering: The View From Here. *Natural Language Engineering*, 7(4):275–300.

Hirschman, L., Light, M., Breck, E., and Burger, J. D. (1999). Deep Read: A Reading Comprehension System. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 325–332, College Park, MD.

Hockenmaier, J. (2003a). *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh.

Hockenmaier, J. (2003b). Parsing with generative models of predicate-argument structure. In *Proceedings of 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 359–366, Sapporo, Japan.

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

Jijkoun, V. and de Rijke, M. (2004). Enriching the Output of a Parser Using Memory-based Learning. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL04)*, pages 311–318, Barcelona, Spain.

Johnson, M. (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.

Judge, J., Cahill, A., Burke, M., O'Donovan, R., van Genabith, J., and Way, A. (2005). Strong Domain Variation and Treebank-Induced LFG Resources. In *Proceedings of the Tenth International Conference on LFG (LFG05)*, pages 186–204, Bergen, Norway.

Judge, J., Cahill, A., and van Genabith, J. (2006). Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 497–504, Sydney, Australia. Association for Computational Linguistics.

Kaplan, R. and Bresnan, J. (1982). Lexical Functional Grammar, a Formal System for Grammatical Representation. In Bresnan, J., editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.

Katz, B., Borchardt, G., and Felshin, S. (2005). Syntactic and Semantic Decomposition Strategies for Question Answering from Multiple Resources. In *Proceedings of the AAAI 2005 Workshop on Inference for Textual Question Answering*, pages 35–41, Pittsburgh, PA.

Kucera, H. and Nelson, F. W. (1967). *Computational Analysis of Present-Day American English.* Brown University Press, Providence, RI.

Lappin, S., Golan, I., and Rimon, M. (1989). Computing Grammatical Functions from Configurational Parse Trees. Technical Report 88.268, IBM Israel, Haifa, Israel.

Levy, R. and Manning, C. (2004). Deep Dependencies from Context-Free Statistical Parsers: Correcting the Surface Dependency Approximation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL04)*, pages 327–324, Barcelona, Spain.

Li, X. and Roth, D. (2002). Learning Question Classifiers. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 556–562, Taipei, Taiwan.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 110–115, Princton, NJ.

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

McCarthy, M. (2003). Design and Evaluation of the Linguistic Basis of an Automatic F-Structure Annotation Algorithm for the Penn-II Treebank. Master's thesis, School of Computing, Dublin City University, Dublin, Ireland.

Miyao, Y., Ninomiya, T., and Tsujii, J. (2003). Probabilistic modeling of argument structures including non-local dependencies. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 285–291, Borovets, Bulgaria.

Noreen, E. W. (1989). *Computer Intensive Methods for Testing Hypotheses: An Introduction.* John Wiley & Sons, New York, NY.

O'Donovan, R., Burke, M., Cahill, A., van Genabith, J., and Way, A. (2004). Large-Scale Induction and Evaluation of Lexical Resources from the Penn-II Treebank . In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 368–375, Barcelona, Spain.

Pasca, M. and Harabagiu, S. M. (2001). High Performance Question/Answering. In *Research and Development in Information Retrieval*, pages 366–374, New Orleans, LA.

Sadler, L., van Genabith, J., and Way, A. (2000). Automatic F-Structure Annotation from the AP Treebank. In Butt, M. and King, T. H., editors, *Proceedings of the Fifth International Conference on LFG*, pages 226–243, Stanford, CA. CSLI Publications.

Santorini, B. (1990). Part-of-Speech guidelines for the Penn Treebank Project. Technical report, University of Pennsylvania, Philadelphia, PA.

Schmid, H. (2000). LoPar: Design and Implementation. Arbeitspapiere des Sonderforschungsbereiches 340, No. 149, IMS Stuttgart.

Silverstein, C., Henzinger, M., Marais, H., and Moricz, M. (1998). Analysis of a Very Large AltaVista Query Log. Technical Report 1998-014, Digital SRC. http://gatekeeper.dec.com/pub/DEC/SRC/technical-notes/abstracts/src-tn-1998-014.html.

Simmons, R. F. (1965). Answering English Questions by Computer: A Survey. *Communications of the ACM*, **8**(1):53–70.

Smeaton, A. F., Over, P., Costello, C. J., de Vries, A. P., Doermann, D., Hauptmann, A., Rorvig, M. E., Smith, J. R., and Wu, L. (2002). Research and Advances Technology for Digital Technology. In M. Agosti, C. T., editor, *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries*, pages 266–275, Rome, Italy.

Voorhees, E. M. (2001). The TREC Question Answering Track. *Natural Language Engineering*, 7(04):361–378.

Waldinger, R. J., Appelt, D. E., Fry, J. S., Israel, D. J., Jarvis, P. A., Martin, D. L., Riehemann, S. Z., Stickel, M. E., Tyson, M., Hobbs, J. R., and Dungan, J. L. (2004). Deductive Question Answering from Multiple Resources. In *New Directions in Question Answering, Mark T Maybury (ed.)*, pages 253–262, Menlo Park, CA. AAAI.

Woods, W. (1973). Progress in Natural Language Understanding - An Application to Lunar Geology. In *AFIPS Conference Proceedings*, pages 441–450, New York, NY.

Appendix A

Penn-II Treebank Tags and Functional Labels

These tagging and functional annotation labels are taken from "Bracketing Guidelines for Treebank II Style Penn Treebank Project" Bies et al. (1995). Reproduced from http://bulba.sdsu.edu/jeanette/thesis/PennTags.html

A.1 Bracket Labels

A.1.1 Clause Level

S simple declarative clause, i.e. one that is not introduced by a (possible empty) subordinating conjunction or a wh-word and that does not exhibit subject-verb inversion.

SBAR Clause introduced by a (possibly empty) subordinating conjunction.

SBARQ Direct question introduced by a wh-word or a wh-phrase. Indirect questions and relative clauses should be bracketed as SBAR, not SBARQ.

SINV Inverted declarative sentence, i.e. one in which the subject follows the tensed verb or modal.

SQ Inverted yes/no question, or main clause of a wh-question, following the wh-phrase in SBARQ.

A.1.2 Phrase Level

ADJP Adjective Phrase.

ADVP Adverb Phrase.

CONJP Conjunction Phrase.

FRAG Fragment.

INTJ Interjection. Corresponds approximately to the part-of-speech tag UH.

LST List marker. Includes surrounding punctuation.

NAC Not a Constituent; used to show the scope of certain prenominal modifiers within an NP.

NP Noun Phrase.

NX Used within certain complex NPs to mark the head of the NP. Corresponds very roughly to N-bar level but used quite differently.

PP Prepositional Phrase.

PRN Parenthetical.

PRT Particle. Category for words that should be tagged RP.

QP Quantifier Phrase (i.e. complex measure/amount phrase); used within NP.

RRC Reduced Relative Clause.

UCP Unlike Coordinated Phrase.

VP Vereb Phrase.

WHADJP Wh-adjective Phrase. Adjectival phrase containing a wh-adverb, as in how hot.

WHAVP Wh-adverb Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing a wh-adverb such as how or why.

WHNP Wh-noun Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing some wh-word, e.g. who, which book, whose daughter, none of which, or how many leopards.

WHPP Wh-prepositional Phrase. Prepositional phrase containing a wh-noun phrase (such as of which or by whose authority) that either introduces a PP gap or is contained by a WHNP.

X Unknown, uncertain, or unbracketable. X is often used for bracketing typos and in bracketing the...the-constructions.

A.1.3 Word level

CC Coordinating conjunction

CD Cardinal number

DT Determiner

EX Existential there

FW Foreign word

IN Preposition or subordinating conjunction

JJ Adjective

JJR Adjective, comparative

JJS Adjective, superlative

LS List item marker

MD Modal

NN Noun, singular or mass

NNS Noun, plural

NNP Proper noun, singular

NNPS Proper noun, plural

PDT Predeterminer

POS Possessive ending

PRP Personal pronoun

PRP$ Possessive pronoun (prolog version PRP-S)

RB Adverb

RBR Adverb, comparative

RBS Adverb, superlative

RP Particle

SYM Symbol

TO to

UH Interjection

VB Verb, base form

VBD Verb, past tense

VBG Verb, gerund or present participle

VBN Verb, past participle

VBP Verb, non-3rd person singular present

VBZ Verb, 3rd person singular present

WDT Wh-determiner

WP Wh-pronoun

WP$ Possessive wh-pronoun (prolog version WP-S)

WRB Wh-adverb

A.2 Function tags

A.2.1 Form/function discrepancies

ADV (adverbial) marks a constituent other than ADVP or PP when it is used adverbially (e.g. NPs or free ("headless" relatives). However, constituents that themselves are modifying an ADVP generally do not get -ADV. If a more specific tag is available (for example, -TMP) then it is used alone and -ADV is implied. See the Adverbials section.

NOM (nominal) marks free ("headless") relatives and gerunds when they act nominally.

A.2.2 Grammatical role

DTV (dative) marks the dative object in the unshifted form of the double object construction. If the preposition introducing the "dative" object is for, it is considered benefactive (-BNF). -DTV (and -BNF) is only used after verbs that can undergo dative shift.

LGS (logical subject) is used to mark the logical subject in passives. It attaches to the NP object of by and not to the PP node itself.

PRD (predicate) marks any predicate that is not VP. In the do so construction, the so is annotated as a predicate.

PUT marks the locative complement of put.

SBJ (surface subject) marks the structural surface subject of both matrix and embedded clauses, including those with null subjects.

TPC ("topicalized") marks elements that appear before the subject in a declarative sentence, but in two cases only:

1. if the front element is associated with a *T* in the position of the gap.
2. if the fronted element is left-dislocated (i.e. it is associated with a resumptive pronoun in the position of the gap).

VOC (vocative) marks nouns of address, regardless of their position in the sentence. It is not coindexed to the subject and not get -TPC when it is sentence-initial.

A.2.3 Adverbials

BNF (benefactive) marks the beneficiary of an action (attaches to NP or PP). This tag is used only when (1) the verb can undergo dative shift and (2) the prepositional variant (with the same meaning) uses for. The prepositional objects of dative-shifting verbs with other prepositions than for (such as to or of) are annotated -DTV.

DIR (direction) marks adverbials that answer the questions "from where?" and "to where?" It implies motion, which can be metaphorical as in "...rose 5 pts. to 57-1/2" or "increased 70

EXT (extent) marks adverbial phrases that describe the spatial extent of an activity. -EXT was incorporated primarily for cases of movement in financial space, but is also used in analogous situations elsewhere. Obligatory complements do not receive -EXT. Words such as fully and completely are absolutes and do not receive -EXT.

LOC (locative) marks adverbials that indicate place/setting of the event. -LOC may also indicate metaphorical location. There is likely to be some varation in the use of -LOC due to differing annotator interpretations. In cases where the annotator is faced with a choice between -LOC or -TMP, the default is -LOC. In cases involving SBAR, SBAR should not receive -LOC. -LOC has some uses that are not adverbial, such as with place names that are adjoined to other NPs and NAC-LOC premodifiers of NPs. The special tag -PUT is used for the locative argument of put.

MNR (manner) marks adverbials that indicate manner, including instrument phrases.

PRP (purpose or reason) marks purpose or reason clauses and PPs.

TMP (temporal) marks temporal or aspectual adverbials that answer the questions when, how often, or how long. It has some uses that are not strictly adverbial, auch as with dates that modify other NPs at S- or VP-level. In cases of apposition involving SBAR, the SBAR should not be labeled -TMP. Only in "financialspeak," and only when the dominating PP is a PP-DIR, may temporal modifiers be put at PP object level. Note that -TMP is not used in possessive phrases.

A.3 Miscellaneous

CLR (closely related) marks constituents that occupy some middle ground between arguments and adjunct of the verb phrase. These roughly correspond to "predication adjuncts", prepositional ditransitives, and some "phrasal verbs". Although constituents marked with -CLR are not strictly speaking complements, they are treated as complements whenever it makes a bracketing difference. The precise meaning of -CLR depends somewhat on the category of the phrase.

- on S or SBAR - These categories are usually arguments, so the -CLR tag indicates that the clause is more adverbial than normal clausal arguments. The most common case is the infinitival semi-complement of use, but there are a variety of other cases.

- on PP, ADVP, SBAR-PRP, etc - On categories that are ordinarily interpreted as (adjunct) adverbials, -CLR indicates a somewhat closer relationship to the verb. For example:

  Prepositional Ditransitives In order to ensure consistency, the Treebank recognizes only a limited class of verbs that take more than one complement (-DTV and -PUT and Small Clauses) Verbs that fall outside these classes (including most of the prepositional ditransitive verbs in class

  $$D2$$

) are often associated with -CLR.

  Phrasal verbs Phrasal verbs are also annotated with -CLR or a combination of -PRT and PP-CLR. Words that are considered borderline between particle and adverb are often bracketed with ADVP-CLR.

  Predication Adjuncts Many of Quirk's predication adjuncts are annotated with -CLR.

- on NP - To the extent that -CLR is used on NPs, it indicates that the NP is part of some kind of "fixed phrase" or expression, such as take care of. Variation is more likely for NPs than for other uses of -CLR.

-CLF (cleft) marks it-clefts ("true clefts") and may be added to the labels S, SINV, or SQ.

HLN (headline) marks headlines and datelines. Note that headlines and datelines always constitute a unit of text that is structurally independent from the following sentence.

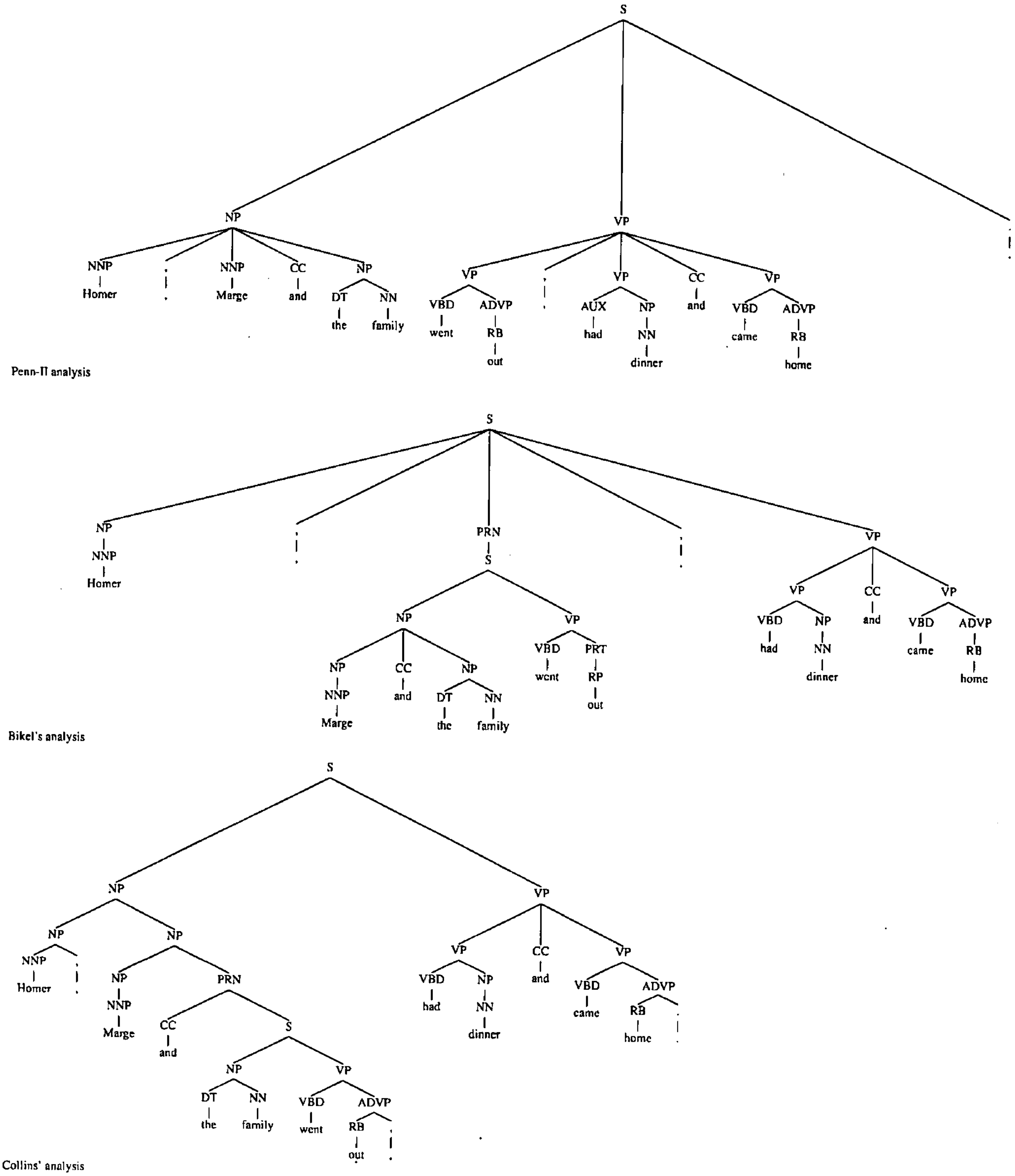
TTL (title) is attached to the top node of a title when this title appears inside running text. -TTL implies -NOM. The internal structure of the title is bracketed as usual.

Appendix B

Additional Figures and Tables

	≤40 words			
	Sect 00		Sect 23	
Model	LR	LP	LR	LP
Collins Model 2	n/a	n/a	88.5	88.7
Bikel's Model 2 emulation	90.0	90.2	88.7	88.9
	All Sentences			
Model	Sect 00		Sect 23	
Collins Model 2	n/a	n/a	88.1	88.3
Bikel's Model 2 emulation	88.8	89.0	88.2	88.3

Table B.1: Comparison of parsing results for Collins' Model 2 parser and Bikel's emulation of Collins' Model 2. (Results are taken from Collins (1999) and Bikel (2004))



Penn-II analysis

Bikel's analysis

Collins' analysis

Figure B.1: Penn-II Punctuation attachment compared with parser output