Open-Domain Question Answering

Mark Andrew Greenwood

Submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

Department of Computer Science
University of Sheffield, UK
September 2005

Dedicated to the memory of
David Lowndes (1979-2003)

"Life is hard somehow"
C & R Macdonald (1999)

Table of Contents

II How To Answer Questions

III How To Define A Term

IV Where Do We Go From Here?

List of Tables

List of Figures

Abstract

Question answering aims to develop techniques that can go beyond the retrieval of relevant documents in order to return exact answers to natural language questions, such as *"How tall is the Eiffel Tower?"*, *"Which cities have a subway system?"*, and *"Who is Alberto Tomba?"*. Answering natural language questions requires more complex processing of text than employed by current information retrieval systems. A number of question answering systems have been developed which are capable of carrying out the processing required to achieve high levels of accuracy. However, little work has been reported on techniques for quickly finding exact answers.

This thesis investigates a number of novel techniques for performing open-domain question answering. Investigated techniques include: manual and automatically constructed question analysers, document retrieval specifically for question answering, semantic type answer extraction, answer extraction via automatically acquired surface matching text patterns, principled target processing combined with document retrieval for definition questions, and various approaches to sentence simplification which aid in the generation of concise definitions.

The novel techniques in this thesis are combined to create two end-to-end question answering systems which allow answers to be found quickly. AnswerFinder answers factoid questions such as *"When was Mozart born?"*, whilst Varro builds definitions for terms such as *"aspirin"*, *"Aaron Copland"*, and *"golden parachute"*. Both systems allow users to find answers to their questions using web documents retrieved by Google™. Together these two systems demonstrate that the techniques developed in this thesis can be successfully used to provide quick effective open-domain question answering.

Preface

Since before the dawn of language humans have hungered after knowledge. We have explored the world around us, asking questions about what we can see and feel. As time progressed we became more and more interested in acquiring knowledge; constructing libraries to hold a permanent record of our ever expanding knowledge and founding schools and universities to teach each new generation things their forefathers could never have imagined. From the walls of caves to papyrus, from clay tablets to the finest parchment we have recorded our thoughts and experiences for others to share. With modern computer technology it is now easier to access that information than at any point in the history of human civilization.

When the World Wide Web (WWW) exploded on to the scene, during the late 80's and early 90's, it allowed access to a vast amount of predominately unstructured electronic documents. Effective search engines were rapidly developed to allow a user to find a 'needle' in this 'electronic haystack'.

The continued increase in the amount of electronic information available shows no sign of abating, with the WWW effectively tripling in size between the years 2000 and 2003 to approximately 167 terabytes of information (Lyman and Varian, 2003). Although modern search engines are able to cope with this volume of text, they are most useful when a query returns only a handful of documents which the user can then quickly read to find the information they are looking for. It is, however, becoming more and more the case that giving a simple query to a modern search engine will result in hundreds if not thousands of documents being returned; more than can possibly be searched by hand – even ten documents is often too many for the time people have available to find the information they are looking for. Clearly a new approach is needed to allow easier and more focused access to this vast store of information.

With this explosive growth in the number of available electronic documents we are entering an age where effective question answering technology will become an essential part of everyday life. In an ideal world a user could ask a question such as *"What is the state flower of Hawaii?"*, *"Who was Aaron Copland?"* or *"How do you cook a Christmas Pudding?"*, and instead of being presented with a list of possibly relevant documents, question answering technology would simply return the answer or answers to the questions, with a link back to the most relevant documents for those users who want further information or explanation.

The Gigablast[1] web search engine has started to move towards question answering with the introduction of what it refers to as *Giga bits* – essentially these *Giga bits* are concepts which are related to the user's search query. For example, in response to the search query *"Who invented the barometer?"* Gigablast, as well as returning possibly relevant documents, lists a number of concepts which it believes may answer the question. The first five of these (along with a confidence level) are Torricelli (80%), mercury barometer (64%), Aneroid Barometer (63%), Italian physicist Evangelista Torricelli (54%) and 1643 (45%). Whilst the first Giga bit is indeed the correct answer to the question it is clear that many of the other concepts are not even of the correct semantic type to be answers to the question. Selecting one of these Giga bits does not result in a single document justifying the answer but rather adds the concept to the original search query in the hope that the documents retrieved will be relevant to both the question and answer. While this approach seems to be a step in the right direction, it is unclear how far using related concepts can move towards full question answering.

One recent addition to the set of available question answering systems, aimed squarely at the average web user, is BrainBoost[2]. BrainBoost presents short sentences as answers to questions; although like most question answering (QA) systems it is not always able to return an answer. From the few implementation details that are available (Rozenblatt, 2003) it appears that BrainBoost works like many other QA systems in that it classifies the questions based upon 'lexical properties' of the expected answer type. This enables it to locate possible answers in documents retrieved using up to four web search engines.

Whilst such systems are becoming more common, none has yet appeared which is capable of returning exact answers to every question imaginable. The natural language processing (NLP) community has experience of numerous techniques which could be applied to the problem of providing effective question answering. This thesis reports the results of research investigating a number of approaches to QA with a view to advancing the current state-of-the-art and, in time, along with the research of many other individuals and organizations, will hopefully lead to effective question answering technology being made available to the millions of people who would benefit from it.

Acknowledgements

Working towards my PhD in the Natural Language Processing (NLP) group at The University of Sheffield has been an enjoyable experience and I am indebted to my supervisor Robert Gaizauskas, not only for his continued support but also for giving me the opportunity to carry out this research. My thanks also to my two advisors, Mark Hepple and Mike Holcombe, for making sure I kept working and for asking those questions aimed at making me think that little bit harder.

Although the NLP research group at Sheffield University is quite large, the number of

[1] `http://www.gigablast.com`

[2] `http://www.brainboost.com`

people actively involved in question answering research is relatively small and I owe a large debt of thanks to all of them for always having the time to discuss my research ideas. They are also the people with whom I have collaborated on a number of academic papers and the annual QA evaluations held as part of the Text Retrieval Conference (TREC); without them this research would not have been as enjoyable or as thorough. They are: Robert Gaizauskas, Horacio Saggion, Mark Hepple, and Ian Roberts.

I owe a huge thank you to all those members of the NLP group who over a number of years have worked hard to develop the GATE framework[3]. Using this framework made it possible for me to push my research further as I rarely had to think about the lower level NLP tasks which are a standard part of the framework.

The TREC QA evaluations have been an invaluable resource of both data and discussion and I am indebted to the organisers, especially Ellen Voorhees.

I'd like to say a special thank you to a number of researchers from around the globe with whom I've had some inspiring conversations which have led to me trying new ideas or approaches to specific problems: Tiphaine Dalmas, Donna Harman, Jimmy Lin, and Matthew Bilotti.

I would also like to thank Lucy Lally for her administrative help during my PhD, I have no idea how I would have made it to most of the conferences I attended without her help.

Of course, none of the research presented in this thesis would have been carried out without the financial support provided by the UK's Engineering and Physical Sciences Research Council[4] (EPSRC) as part of their studentship programme for which I am especially grateful.

Any piece of technical writing as long as this thesis clearly requires a number of people who are willing to proof read various drafts in an effort to remove all the technical and language mistakes. In this respect I would like to thank Robert Gaizauskas, Mark Stevenson, Pauline Greenwood, John Edwards, Angus Roberts, Bryony Edwards, Horacio Saggion, and Emma Barker. I am also grateful to my examiners, Louise Guthrie and John Tait, for their constructive criticism which led to immeasurable improvements in this thesis. I claim full responsibility for any remaining mistakes.

On a more personal note I would like to thank the family and friends who have given me encouragement and provided support during my time at University. I would specifically like to thank my parents without whose help I would never have made it to University in the first place. Their continued support and encouragement has helped me maintain my self-confidence throughout the four years of this research. Without the unwavering support of my fiancée I do not know if I would have made it this far and I am eternally grateful for her belief in my ability to finish this thesis.

[3] `http://gate.ac.uk`

[4] `http://www.epsrc.co.uk/`

Part I

What Is Question Answering?

Chapter 1

Question Answering: An Overview

question ► noun a sentence worded or expressed so as to elicit information.
answer ► noun a thing that is said, written, or done to deal with or as a reaction to a question, statement, or situation.
New Oxford Dictionary of English (2001)

We all know what a question is and often we know what an answer is. If, however, we were asked to explain what questions are or how we go about answering them then many people would have to stop and think about what to say. This chapter gives an introduction to what we mean by question answering and hence the challenges that the approaches introduced in this thesis are designed to overcome.

1.1 Questions

One definition of a question could be 'a request for information'. But how do we recognise such a request? In written language we often rely on question marks to denote questions. However, this clue is misleading as rhetorical questions do not require an answer but are often terminated by a question mark while statements asking for information may not be phrased as questions. For example the question *"What cities have underground railways?"* could also be written as a statement *"Name cities which have underground railways"*. Both ask for the same information but one is a question and one an instruction. People can easily handle these different expressions as we tend to focus on the meaning (semantics) of an expression and not the exact phrasing (syntax). We can, therefore, use the full complexities of language to phrase questions knowing that when they are asked other people will understand them and may be able to provide an answer.

While there are a number of different forms of question, this thesis is primarily concerned

with factoid and definition questions as defined within the QA framework used within the Text Retrieval Conference (TREC, see Section 2.4 for an overview).

Factoid questions are those for which the answer is a single fact. For example *"When was Mozart born?"*, *"How tall is the Eiffel Tower?"*, and *"Where is Microsoft based?"* are all examples of factoid questions. There are many questions which are not considered to be factoid questions and which are not covered by this thesis. These include questions which can have yes/no answers, such as *"Is London's population bigger than that of Paris?"* as well as instruction based questions (e.g. *"How do I make tomato soup?"*) and explanation questions (e.g. *"Why did America enter WWII?"*). There is another type of question referred to as a list question which is closely related to factoid questions. List questions are factoid questions that require more than one answer. For example *"What grapes are used in making wine?"* is a list question. While list questions will not be covered in any detail in this thesis, most of the approaches to factoid question answering can also be used to answer list questions (see Appendices D, E and F for the official TREC results of answering list questions using the factoid techniques developed for this thesis).

Factoid questions have been the main focus of recent QA research partly because they are the main focus of the QA evaluation held annually since 1999 as part of TREC (Voorhees, 1999). Recent evaluations (Voorhees, 2004) suggest that the current state-of-the-art QA systems can answer at most 80% of factoid questions (although the median score in the same evaluation was much lower at only 17%).

Definition questions, unlike factoid questions, require a more complex answer, usually constructed from multiple source documents. The answer should be a short paragraph which succinctly defines the *definiendum* (the thing – be it person, organization, object or event, often referred to as the target) which the user wishes to know more about. Good answers to definition questions should probably be very similar in nature to an entry in an encyclopaedia. For example, if the question asks about a person then the user will probably want to know the important dates in their life (birth, marriage and death), their major achievements and any other interesting items of note. For an organization the definition should probably include information about its purpose, when and by whom it was founded, the number of employees, and other interesting facts according to the nature of the organisation.

Definition questions have also been included in recent TREC QA evaluations, although they are referred to as *other* questions and are treated as meaning *"tell me anything interesting about the target that I have not asked about directly"*. Recent evaluations (Voorhees, 2004) suggest that the state-of-the-art QA systems can achieve an F-measure score of approximately 0.46, with the median score in the same evaluation being approximately 0.18.

1.2 Answers

If a question is a request for information and answers are given in response to questions, then answers must be responses to requests for information. But what constitutes an answer? Almost any statement can be an answer to a question and, in the same way that there can be many different ways of expressing the same question, there can be many ways of describing the same answer. For example, any question whose answer is numeric may have the answer expressed in an infinite number of ways.

While there maybe multiple ways of expressing the correct answer to a question, not all will fulfil the needs of the questioner. For example, if *Stocksbridge* was given in answer to the question *"Where is Deepcar?"*, whilst a correct answer, it will only be useful to someone who knows where *Stocksbridge* is – near Sheffield in the north of England. For the purposes of the evaluations in this thesis we will assume, as do the TREC evaluations, that the questioner is a native speaker of English and an average reader of newspapers.

There have been many attempts to define what constitutes a correct answer produced by a QA system, many of which have centred around the TREC QA evaluations, resulting in answer definitions including:

- In TREC-8 (Voorhees, 1999) an answer was defined as a string of up to 50 or 250 characters in length which contained a correct answer in the context provided by the document;
- For TREC 2003 (Voorhees, 2003b) a response was judged correct if it *"...consists of exactly a right answer and that answer is supported by the document returned."*;
- A number of studies have used the TREC questions and have defined an answer to be a text snippet which matches an answer judged to have been correct in the original evaluation and which comes from a document also judged to have been relevant (Roberts and Gaizauskas, 2004).

A precise, all encompassing, definition of what constitutes an answer is difficult to come by. Whilst an answer has to be correct to be of any use this still leaves a lot of scope for different systems to present the same answer in different ways.

1.3 The Process of Question Answering

If we think about question answering as a human activity then what do we expect to happen when a person is asked a question to which they do not know the answer? In this situation it is likely that the person of whom the question was asked would consult some store of knowledge (a book, a library, the Internet...) in order to find some text that they could read and understand, allowing them to determine the answer to the question. They could then return to the person who had originally asked the question and tell them the

answer. They could also report on where they had found the answer which would allow the questioner to place some level of confidence in the answer.

What they would not do would be simply to give a book or maybe a handful of documents, which they thought might contain the answer, to the person asking the question. Unfortunately this is what most computer users have to be content with at the moment. Many people would like to use the Internet as a source of knowledge in which they could find answers to their questions. Although many search engines suggest that you can ask natural language questions, the results that they return are usually sections of documents which may or may not contain the answer but which do have many words in common with the question. This is because question answering requires a much deeper understanding and processing of text than most web search engines are currently capable of performing.

1.4 The Challenges of Question Answering

So what is it that makes question answering hard? What is it that people do when reading text to find the answer to a question that computers have to be programmed to do so as to be able to function in a similar fashion?

If we sidestep the issue of determining if a question is being asked by assuming that all input to a QA system is in fact a question (the problems of a wider dialogue are beyond the scope of this study although the history of dialogue systems, as they pertain to question answering, is discussed in Chapter 2) then there are a number of issues we have to confront.

There are often many ways of asking for the same piece of information. Sometimes the variations are simple re-wordings of the question and other times the variations can depend on the context in which the question is being asked or the knowledge of the questioner.

In a similar way there are many ways of describing the same answer and so context and user knowledge can play a role in the way answers are defined. For instance an answer of *"last Tuesday"* means nothing without knowledge of the current date or the date on which the source of the answer was written. Not only does this make it difficult for a QA system to determine if two answers are equivalent (for the purposes of combining and ranking answers) it also makes evaluating the output of QA systems problematic (see Chapter 3).

The fact that both questions and answers can be written in many different ways makes it difficult to ensure that documents which discuss the subject of the question are retrieved. For instance, the answer to the question *"What is the capital of France?"* may well be written as *"Paris is the French capital"* within a document. Of course both could also be written as *"What is the French capital?"* and *"Paris is the capital of France"* respectively. Not only does this example show that both questions and answers can be written in multiple ways but also that unless the question and answer are expressed in a similar way there may be very little in common between the question and answer (in this

example only the word ‘capital’ is common to the different forms) making it very difficult to retrieve documents containing the answer.

Whilst questions such as *“What is aspirin?”* (usually referred to as a definition question) show little variation, their answers are usually more varied than for questions which request a single fact, such as a person’s name or a date. Finding relevant documents for such questions is also extremely challenging. The only word(s) available to narrow the search are from the term being defined and words are often used in a document without them first having been defined. This means many documents which mention the term being defined will be of no use to a system attempting to create a definition.

Clearly there are many challenges in developing computer systems capable of answering questions. This thesis will focus on two particular types of question; usually referred to as factoid and definition (Parts II and III respectively). Each question will be self-contained and asked in isolation. This of course removes from consideration some of the issues previously discussed, especially that of context, leaving us free to concentrate on the remaining issues.

1.5 Thesis Aims and Objectives

The research documented in this thesis was not undertaken with the sole aim of producing a highly effective question answering system, but rather to explore some of the challenges currently facing open-domain question answering research in the hope of furthering the understanding of this highly interesting and promising field. While the natural language processing (NLP) community has many techniques which could be applied to question answering (such as syntactic and semantic parsing for a deeper understanding of texts) we should not lose sight of the fact that question answering promises to revolutionize the way in which we access large text collections such as the Internet. As such we must keep the end user of such technologies in mind when pushing the frontiers of research. For instance, a QA system capable of correctly answering any question put to it would be of limited practical use if it required an hour to answer each question. People are used to fast responses when using web search engines and whilst they may be willing to wait twice as long for an exact answer compared to a selection of documents this would still mean that QA systems would have only seconds in which to answer a question. For this reason the research documented in this thesis is focused on techniques to analyse questions, retrieve relevant documents, and extract exact answers that can be used in both building effective QA systems and increasing our understanding of the subject.

The main motivation behind the work in this thesis was a growing frustration experienced with a previous question answering system, QA-LaSIE (Greenwood et al., 2002). QA-LaSIE is a large and complex QA system using full semantic parsing to analyse questions and documents in order to locate candidate answers. This complexity allows the system to answer a wide range of questions types but it makes understanding or changing the inner workings of the system extremely time consuming. A further side affect of this complexity is that it takes minutes (and sometimes hours) to answer a single question.

This does not affect the accuracy of the system, however, it does limit it's use in real-world applications. The techniques developed in this thesis, therefore, take alternative approaches to those used in QA-LaSIE. Where possible, simpler techniques have been developed in order to a) reduce the overall system complexity and b) result in systems which can quickly answer questions and could therefore form the basis of useful real-world applications. With these aims in mind, we introduce novel approaches to a number of different aspects of question answering.

Our approaches for answering factoid questions rely on a three component architecture of question analysis, document retrieval, and answer extraction. For question analysis we introduce a rule formalism for constructing hand crafted question classifiers, and a high performance question classifier which uses a k-Nearest Neighbours style algorithm in conjunction with an information retrieval (IR) engine. We introduce a number of different approaches to improve document retrieval specifically for QA. These include various approaches to query formulation, such as pertainym expansion, as well as experiments looking into the amount of text to retrieve for question answering and comparisons between retrieval from closed collections and the web. Two approaches to answer extraction are introduced and described in detail. The first of the two approaches builds upon the manually constructed question classifier and is based upon semantic type extraction. The second approach revolves around automatically acquiring generalized surface matching text patterns.

We also introduce an approach to answering definition questions which includes novel techniques for the analysis of definition targets, filters for sentence selection, and redundant phrase detection. Together, these techniques lead to a high performance definitional question answering system.

Whilst a number of different techniques for answering factoid and definition questions are introduced and evaluated throughout this thesis it should be remembered that a long term aim is to produce approaches which can be used in real-world QA applications. Two publicly available QA systems are introduced which combine the techniques introduced throughout this thesis: AnswerFinder and Varro.

AnswerFinder, detailed in Chapter 9, combines the approaches to factoid question answering discussed earlier in the thesis into an application which draws its answers from the web. On average, AnswerFinder takes approximately 8 seconds to answer a question – not an unreasonable length of time to wait for an exact answer to a question. Optimization, which has not been performed, could certainly improve upon this.

Varro, detailed in Appendix C, constructs a definitions for a given target using Google to search the web for relevant documents from which it extracts sentences that describe some aspect of the target. On average definitions are created in approximately 21 seconds. While this is much longer than people are usually willing to wait for a search engine to locate relevant documents it should be compared with the time it would take a user to read the documents to find the interesting facts about the target. Seen in this light 21 seconds seems reasonable and code optimizations could improve upon this.

Many of the approaches developed throughout this thesis have been evaluated within the QA evaluations held as part of TREC. The performance of the approaches for answering factoid, list and definition questions was above the average of participating systems when independently evaluated as part of TREC 2004 and TREC 2005 (see Appendices D, E and F).

1.6 Thesis Structure

This thesis consists of four main sections which answer the aims stated above. Part I presents the relevant background material and contains a broad overview of questions and answers along with a brief history of computer based question answering. Most of this opening section is devoted to detailing the numerous evaluation frameworks and metrics required to allow us both to evaluate the research contained in later sections of this thesis, and to compare the results with the work of other researchers in the field.

Part II contains novel contributions to the field of open-domain factoid question answering and is divided into chapters that deal with the three main components of most modern QA systems: question analysis, document retrieval and answer extraction.

Part III is organized in a similar fashion to Part II and details novel contributions to definitional question answering.

Finally, Part IV attempts to sum up the ideas presented in this thesis as well as suggesting possible future directions for research into QA technology.

Chapter 2

A Brief History of QA

It would be wrong to claim that interest in QA technology is a recent development in Natural Language Processing (NLP) with Simmons (1965) having reviewed no less than fifteen English language QA systems constructed between 1960 and 1965. This chapter will present a brief history of QA in an attempt to place the research detailed in Parts II and III of this thesis in the proper historical context.

2.1 Natural Language Database Systems

Two of the best-known early QA systems were BASEBALL (Green et al., 1961) and LUNAR (Woods, 1973). The BASEBALL system was designed to answer questions about baseball games which had been played in the American league during a single season, while LUNAR was designed *"...to enable a lunar geologist to conveniently access, compare and evaluate the chemical analysis data on lunar rock and soil composition that was accumulating as a result of the Apollo moon mission"* (Woods, 1973). Both systems were much more than toy research projects, with LUNAR being successfully demonstrated at the Second Annual Lunar Science Conference in 1971. Of the 111 questions that were non-comparative, and within the scope of the moon rock data, 78% were answered correctly.

Although many of these early systems were highly sophisticated even by modern standards, they were nearly all restricted to a limited domain with access to a structured database containing the available domain knowledge. The questions presented to these systems were usually analysed using linguistic knowledge to produce a canonical form, which was then used to construct a standard database query. For example, for the question[1] *"List the authors who have written books about business"* an SQL (Structured Query Language) query, such as the following, would be generated:

```
SELECT firstname, lastname FROM authors, titleauthor, titles
```

[1] This is a modern example taken from Microsoft's English Query 2000, which is part of Microsoft SQL Server. See `http://www.microsoft.com/sql/` for more information.

```
WHERE authors.au_id = titleauthor.au_id
AND titleauthor.title_id = titles.title_id
```

In simple terms, these early systems relied on having the knowledge required to answer a question available in a highly structured form, not as completely unstructured text, which is one of the challenges facing today's QA researchers.

Most QA research that took place during the 1970's was in a similar vein to the systems already mentioned, more examples of which can be found in Grosz et al. (1986). See Copestake and Spärck Jones (1990) for a comprehensive review of natural language front-end development through to the year 1990.

While it appears that little research into QA as an independent task was being undertaken, many of the early research projects were concerned with related tasks that would form the basis of future QA research. Such related ideas are dialogue and reading comprehension, both of which are discussed below with the aim of highlighting their contributions to open-domain QA.

2.2 Dialogue Systems

In his seminal article *"Computing Machinery and Intelligence"*, Turing (1950) described an experiment aimed at settling the question of whether or not a computer is capable of thought. The Turing Test, as the experiment has become known, gave birth to research into computer systems capable of holding a meaningful conversation, usually referred to as a dialogue. A dialogue, by its very nature, often contains at least one question and therefore is relevant to this history of QA development.

One of the earliest and best known of these Artificial Intelligence (AI) dialogue systems is Weizenbaum's ELIZA (1966). ELIZA was designed to emulate a therapist, and for many years could be found listening to visiting academics in the AI Laboratory at the Massachusetts Institute of Technology (M.I.T.). Although on first sight ELIZA seems extremely sophisticated, a closer look at how the program works shows just how simplistic it really is. ELIZA operates through sequences of pattern matching and string replacement, for example the pattern

... you ... me

matches the sentence

Why do you hate me?

On encountering this sentence the program selects a transformation from the list of possible transformations for this pattern, for example:

YOU LIKE TO THINK I ... YOU DON'T YOU?

and produces the output

YOU LIKE TO THINK I HATE YOU DON'T YOU?

Clearly this is not a robust dialogue system and there are many examples of ELIZA producing complete gibberish due to stringently applying these transformation rules. It should also be clear that this type of system is never going to be able to answer open-

domain questions which require a real-world response, such as *"What do penguins eat?"*. For a complete exposé of ELIZA and a similar program called PARRY (Heiser et al., 1980), see Copeland (1993).

Moving forward through the history of AI dialogue systems research brings us to SHRDLU (Winograd, 1972) and GUS (Bobrow et al., 1977). Both of these systems are interfaces to databases holding information about a restricted domain. The difference between these systems and systems such as LUNAR (see Section 2.1) are their dialogue capabilities. GUS was designed to simulate a travel advisor and had access to a database containing limited information about airline flight times.

SHRDLU is probably the better known of these two systems. It controlled a robot arm in a virtual micro-world consisting of a table top strewn with coloured blocks of varying shapes and sizes and a box into which the blocks could be placed[2]. Whilst example conversations with SHRDLU are generally impressive, the system is still severely limited to only discussing the micro-world it inhabits. Take the following exchange[3] (this is a slightly modified version of a conversation, due to Haugland (1985), which appeared in Copeland (1993)):

> *Build a steeple.*
> SORRY, I DON'T KNOW THE WORD 'STEEPLE'.
> *A 'steeple' is a stack that contains two green cubes and a pyramid.*
> I UNDERSTAND.
> *Trade you the steeple for three red cubes.*
> SORRY, I DON'T KNOW THE WORD 'TRADE'.
> *A 'trade' is a free exchange of ownership.*
> SORRY, I DON'T KNOW THE WORD 'FREE'.
> *Sorry, I thought you were smarter than you are.*
> SORRY, I DON'T KNOW THE WORD 'SORRY'.

This attack is a little unfair, as no one ever claimed that the system had any knowledge about trade or anything other than the properties of the blocks, table, box and the ability to move the blocks around. In fact even colours, which SHRDLU seems to understand really well, are simply properties of the blocks; there is no understanding of colour. Although an outstanding programming effort, SHRDLU is as limited as ELIZA and in its creator's own words *"a dead end"*.

Dialogue systems were historically the domain of AI researchers. This has changed over time, and currently there is a vast amount of NLP-based research into dialogue systems. One modern dialogue system is Jupiter (Zue et al., 2000), probably best described by its product page at MIT[4]:

[2] `http://hci.stanford.edu/cs147/examples/shrdlu/`

[3] Note that in the machine-human transcripts the participants are a *human* and a MACHINE.

[4] `http://www/sls.lcs.mit.edu/sls/whatwedo/applications/jupiter.html`

> *"Jupiter is a conversational system that provides up-to-date weather information over the phone. Jupiter knows about 500+ cities worldwide (of which 350 are within the US) and gets its data from four different Web-based sources".*

The following are example questions put to the Jupiter system; note how the system remembers some aspects of the previous queries:

- What cities do you know about in California?
- How about in France?
- What will the temperature be in Boston tomorrow?
- What about the humidity?
- Are there any flood warnings in the United States?
- Where is it sunny in the Caribbean?
- What's the wind speed in Chicago?
- How about London?
- Can you give me the forecast for Seattle?
- Will it rain tomorrow in Denver?

Jupiter is based on the GALAXY client-server architecture (Seneff et al., 1998; Polifroni and Seneff, 2000) and consists of the following stages:

1. Speech Recognition: converts the spoken sentence into text.
2. Language Understanding: parses the text into semantic frames (grammatical structures containing the basic terms need to query the Jupiter database).
3. Language Generation: uses the semantic frame's basic terms to build a SQL query for the database.
4. Information Retrieval: Jupiter executes the SQL query and retrieves the requested information from the database.
5. Language Generation: converts the query result into a natural language sentence.
6. Information Delivery: Jupiter delivers the generated sentence to the user via voice (using a speech synthesizer) and/or display.

Clearly Jupiter is more complex than systems such as SHRDLU as the system is dealing with input via the telephone and hence has to cope with the added problem of robust speech recognition to provide a reasonable input to the dialogue system. Note, however, that just as SHRDLU was limited to questions about the block world it inhabited so Jupiter is limited to questions about weather reports for the cities it is aware of.

2.3 Reading Comprehension Systems

Reading comprehension tests are frequently used to test the reading level of children. As early as 1977 (see the QUALM system detailed in (Lehnert, 1977)) researchers recognised that they were also an appropriate way of testing the language understanding abilities of computer programmes and hence started to build systems capable of answering reading comprehension tests.

In a similar way to most of the early dialogue systems (see the previous section), the early reading comprehension systems were developed by the AI community. One of the best known early systems is SAM, developed at the Yale AI lab, and which arose from earlier work on inference and paraphrase in the MARGIE system (Schank, 1975). Consider the following short story:

> *John went to New York by bus. On the bus he talked to an old lady. When he left the bus, he thanked the driver. He took the subway to Leone's. On the subway his pocket was picked. He got off the train and entered Leone's. He had some lasagne. When the check came, he discovered he couldn't pay. The management told him he would have to wash dishes. When he left, he caught a bus to New Haven.*

This is a simple story and one about which we could all answer simple questions. The following is the transcript of a set of questions about the above story being put to SAM (Schank and Abelson, 1977):

> *Why did John go to New York?*
> BECAUSE JOHN WANTED TO GO TO LEONE'S.
> *How did John get to Leone's?*
> JOHN TOOK A BUS TO NEW YORK AND THEN HE TOOK THE SUBWAY TO LEONE'S.
> *Did anything unusual happen on the bus?*
> I DON'T REMEMBER ANYTHING.
> *Did anything unusual happen on the subway?*
> A THIEF PICKED JOHN'S POCKET.
> *What did John order?*
> JOHN ORDERED LASAGNE.
> *Did anything unusual happen at the restaurant?*
> JOHN DISCOVERED THAT HE COULDN'T PAY THE CHECK AND SO HE HAS TO WASH DISHES.
> *Why couldn't John pay the check?*
> BECAUSE JOHN DID NOT HAVE ANY MONEY.
> *Why didn't John have any money?*
> BECAUSE A THIEF PICKED JOHN'S POCKET.
> *How did John get back to New Haven?*
> JOHN WENT FROM LEONE'S BY SUBWAY AND THEN HE WENT TO NEW HAVEN

BY BUS.
How was John able to take the bus to New Haven?
PROBABLY JOHN HAD A TICKET.

This is without doubt an impressive display. Of special interest is the fact that SAM has managed to return information which is only implicitly stated in the story, for example:

- SAM inferred that John ordered the lasagne that he had
- SAM inferred that having to wash the dishes was a consequence of not being able to pay the check
- SAM manages to correctly decide that John's reason for going to New York was to eat in Leone's not to wash dishes or have his pocket picked.
- In the answers to the last two questions SAM makes sensible conjectures well above the information given in the story.

This dazzling display is all the work of numerous scripts which SAM applies as it works through a story (incidentally, SAM stands for *Script Applier Mechanism*). In this instance SAM would use scripts for restaurant, bus and subway. These scripts allow simple stories to be expanded to contain all the standard things that happen in a situation (such as sitting at a table in a restaurant although that is never mentioned). Knowing exactly what should happen in a restaurant enables SAM to spot deviations from the norm, i.e. in this case John is unable to pay the check. Having already applied the subway script and noticing that the usual outcome of having your pocket picked is no money, SAM can then correctly deduce that John cannot pay the check because he has no money. Like many of the systems (in numerous domains) which we have already discussed SAM is limited in that a script must exist for SAM to sensibly answer any questions. Clearly there will come a time when a script is needed which has not been prepared and the system will fail. The aim of this type of research must then be to get away from the necessity of hand-coded resources, to open-domain unrestricted question answering (the same problem that haunted early dialogue processing systems).

Many of the modern reading comprehension systems are designed to return only the sentence most likely to contain the answer, and not just the answer itself. Although this is a step backwards compared to systems such as SAM, this limitation is partly based on the fact that these systems no longer rely on scripts to generate answers. This contrasts with most other question answering research in which systems aim to return an answer (albeit surrounded by text from within a sentence) rather than the full sentence containing the answer. Two such systems are Quarc (Riloff and Thelen, 2000) and Deep Read (Hirschman et al., 1999) both of which report results at between 30% and 40% in reading comprehension tests for children in the 3rd to 6th grades[5]. An example test is shown in Figure 2.1.

[5] For those not familiar with the grade school system, children in these grades are between eight and twelve years old.

How Maple Syrup is Made

Maple syrup comes from sugar maple trees. At one time, maple syrup was used to make sugar. This is why the tree is called a "sugar" maple tree. Sugar maple trees make sap. Farmers collect the sap. The best time to collect sap is in February and March. The nights must be cold and the days warm. The farmer drills a few small holes in each tree. He puts a spout in each hole. Then he hangs a bucket on the end of each spout. The bucket has a cover to keep rain and snow out. The sap drips into the bucket. About 10 gallons of sap come from each hole.

1. Who collects maple sap? (Farmers)
2. What does the farmer hang from a spout? (A bucket)
3. When is sap collected? (February and March)
4. Where does the maple sap come from? (Sugar maple trees)
5. Why is the bucket covered? (to keep rain and snow out)

Figure 2.1: An example reading comprehension test.

Both systems work by using a set of pattern matching rules (often just bag-of-words) and then augmenting this with one or more of the following natural language techniques: part of speech (POS) tagging, stemming, named entity identification, semantic class identification and pronoun resolution.

At first glance these systems seem exceptionally poor when compared to other QA systems, such as those entered in TREC, which at best answer approximately 70% of the questions. As was pointed out by Anand et al. (2000), reading comprehension tests are actually document-specific question answering tasks:

> *"Each question is asked with respect to a specific document and the answer must be located from within that document ... document-specific question answering poses different challenges than general question answering because an answer generally appears only once in a document ... whereas in general QA many documents contain an answer to the question, hence a document-specific system usually only has one shot to find the answer."*

One modern system that attempts to return an actual answer, rather than the sentence most likely to contain the answer, is Spot (Anand et al., 2000). Spot was developed around the hypotheses that:

> *"... one can fruitfully decompose the reading comprehension task into question analysis (**QAnalysis**) categorizing the question as one of 30 odd types, finding an answer region (**HotSpotting**), and finding the answer phrase in the answer region (**PinPointing**)"*

The system they then implemented uses this hypothesis to attack the problem as follows:

QAnalysis: categorise the question based on a shallow parse of the question combined with lexically grounded regular expressions.

HotSpotting: find the answer region (i.e. sentence) using word overlap between question and region.

PinPointing (1): use independent tagger modules to mark phrases with types corresponding to the question types from QAnalysis.

PinPointing (2): rank the candidate answers using information from QAnalysis, HotSpotting, and PinPointing (1). Candidate ranking is necessary since HotSpotting and PinPointing cannot be performed perfectly.

Although a valiant effort, they still only produced a system which could answer about 28% of the questions (clearly the result was going to be worse than the systems which just return a sentence as this is a more difficult task). However, if the system is evaluated between the final two stages then the performance is comparable with Quarc and Deep Read.

2.4 Open-Domain Factoid Question Answering

One of the early open-domain question answering systems designed to extract exact answers from free text, rather than a structured database, was MURAX (Kupiec, 1993). MURAX was designed to answer questions from the Trivial Pursuit general-knowledge board game – drawing answers from Grolier's on-line encyclopaedia (1990). Answers were assumed to be noun phrases and evaluation over seventy 'who' and 'what' questions shows that MURAX was able to return a single correct exact answer to 53% of the questions. A correct answer was within the first five answers returned for 74% of the questions. It is difficult to compare the performance of MURAX with more modern QA systems, especially as using an encyclopaedia as the document collection is likely to make finding correct answers easier. For example, when asking the question *"What's the capital of the Netherlands?"* it is likely that the encyclopaedia entry for the Netherlands will be highly ranked and is very likely to contain the correct answer. Asking the same question over a corpus of newswire articles or unrestricted text, such as the web, is likely to be harder.

In recent years research in open-domain QA has been accelerated due to the inclusion of a QA track at the Text Retrieval Conference (TREC). TREC was started in 1992 with the aim of supporting information retrieval research by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies[6]. The QA track was first included as part of TREC in 1999 with seventeen research groups entering one or more systems. Although the performance of the systems varied widely, some were remarkably good (see Voorhees (1999) for an overview of the track and Moldovan et al. (1999) for a report on the best overall system). An additional aim of the track was to define a task that

[6] See `http://trec.nist.gov/overview.html` for a more detailed overview of TREC.

would appeal to both the document retrieval community (as systems could return up to 250 bytes, the task could be seen as short passage retrieval) and the information extraction (IE) community (where question answering is simply open-domain IE).

The majority of the systems work in a similar fashion and consist of two main (often entirely separate) sub-systems. Firstly, an IR system is used to select the top n documents or passages which match a query that has been generated from the question. For more details on this stage in the workings of a question answering system see Chapter 7.

The second stage then consists of finding the answer entities (usually snippets of text) from within these documents and then ranking them in such a way as to select a limited number of possible answers. The majority of the early TREC systems pinpointed likely answers by using a form of window-based word scoring, which rewards desirable words in the window. They moved a window across the candidate answer text and returned the window at the position giving the highest score. Clearly many variations on this technique are available by, for example, tuning the window size and the score assigned to different words. Although this form of answer pinpointing works to some degree (giving results of up to 30% in independent evaluations), it has some serious limitations (Hovy et al., 2001):

- It is impossible to accurately pinpoint the boundaries of an answer (e.g. an exact name or phrase).
- It relies solely on word level information and does not use semantic information (hence no knowledge of the type, i.e. person or location, of the answer being sought).
- It is impossible to see how this method could be extended to composing an answer from many different documents or even from different sentences or phrases within a single document.

Window based answer-pinpointing techniques are therefore limited and will not, in the long run, be a satisfactory method for pinpointing candidate answers. This has led to more and more of the TREC systems implementing semantic methods for pinpointing answers.

Currently the state-of-the-art factoid QA systems are able to answer between 65% and 70% of the TREC questions, with the average being approximately 20% (Voorhees, 2004). Obviously there is a large difference in performance between the best performing systems and the rest of the field. Unfortunately the time taken to answer the TREC questions is not part of the evaluation and so it is unclear if any of the top performing systems are useable in real-time. At TREC 2004, Language Computer Corporation (LCC) (known for developing PowerAnswer – a consistently well performing TREC QA system) introduced a new system called PALANTIR designed to answer questions in under 20 seconds for use in their interactive dialogue system, FERRET. Forcing PALANTIR to return an answer within twenty seconds involves a trade-off between the precision of the answer and the

speed with which the answer is returned (Moldovan et al., 2004). The results of PALANTIR compared with LCC's main QA system, PowerAnswer, clearly show that attempting to return answers quickly has a significant effect on the ability of the system to provide correct answers to the questions (the factoid score dropped from 0.770 to 0.339 and the overall combined score dropped from 0.601 to 0.254 over the TREC 2004 test set). Not only does the performance of the QA system decrease dramatically when forced to return answers quickly but the AQUAINT collection, from which answers must be drawn, had to be processed in advance and indexed via named entities to achieve this (see Section 4.1 for details on the AQUAINT collection).

The future of open-domain question answering as defined by the TREC tasks is likely to be guided by both the roadmap document (Burger et al., 2000) and the ARDA AQUAINT program[7]. The ultimate goal of the AQUAINT program is not to develop QA systems which only answer factually based questions whose answers can be found within a relatively short window of text (e.g. a 50 or 250 byte window) from a single document, but to address a scenario in which multiple, inter-related questions are asked in a focused topic area by a skilled, professional information analyst who is attempting to respond to larger, more complex information needs or requirements. While systems do exist which offer some results in these areas, they are limited and do not meet the US government's broader requirements for question answering. The current suggestion is that participants in the AQUAINT program will attempt harder tasks than the participants in the standard TREC QA track. When these systems are achieving reasonable results the tasks will be moved into the standard QA track.

2.5 Definition Questions

Systems which attempt to answer definition questions differ quite substantially from standard factoid QA systems. This is understandable as the requirements of an answer to a definition question are different to those of a factoid question. Factoid questions require a single fact (or a set of facts for a list question) to be returned to the user. Definition questions require a substantially more complex response – a short paragraph which succinctly defines the *definiendum* (the thing – be it person, organization, object or event, often referred to as the target) which the user wishes to know more about.

Good answers to definition questions should probably be very similar in nature to an entry in an encyclopaedia. For example, if the question asks about a person then important dates in their life (birth, marriage and death), their major achievements and any other interesting items of note would comprise a 'correct' answer. For an organization the answers should probably include information about when and by whom the company was founded, what the company makes/sells, who owns the company, and other interesting things such as other companies they have bought or collaborated with.

TREC 2003 introduced definition questions as a new type of question leading to the first

[7] `http://www.ic-arda/InfoExploit/aquaint/index.html`

large scale evaluation of systems capable of answering these questions. Fifty questions were asked including 30 for which the target was a (possibly fictional) person (e.g. Vlad the Impaler, Ben Hur, Alice Rivlin), 10 for which the target was an organization (e.g. Baush & Lomb, Friends of the Earth) and 10 for which the target was some other thing (e.g. a golden parachute, the medical condition shingles, the Kama Sutra). This evaluation was based around the idea that a definition should contain certain *nuggets* of information, some of which are vital to a good definition whilst others simply expand on it but are not necessary.

One of the problems with systems designed to return definitions is the level of detail the systems should go into when producing an answer. For the TREC evaluations the following scenario is assumed (Voorhees, 2003b):

> *The questioner is an adult, a native speaker of English, and an "average" reader of US newspapers. In reading an article, the user has come across a term that they would like to find out more about. They may have some basic idea of what the term means either from the context of the article (for example, a bandicoot must be a type of animal) or basic background knowledge (Ulysses S. Grant was a US president). They are not experts in the domain of the target, and therefore are not seeking esoteric details (e.g. not a zoologist looking to distinguish the different species in genus Perameles).*

The best way to accurately describe the kind of information about a target that systems were expected to return is by way of an example. The following are the nuggets that systems were expected to return in response to the question *"What is a golden parachute?"*

1. vital – Agreement between companies and top executives
2. vital – Provides remuneration to executives who lose jobs
3. vital – Remuneration is usually very generous
4. okay – Encourages execs not to resist takeover beneficial to shareholders
5. okay – Incentive for execs to join companies
6. okay – Arrangement for which IRS can impose excise tax

From this single example we can see firstly that each nugget is a self-contained piece of information, and secondly that the vital nuggets accurately define the term while the other nuggets simply expand the ideas without adding anything really new to the definition. It should be noted that whilst one person may agree with a set of nuggets another person may have a totally different idea as to how to define a given term. That is, the evaluation of a definitional QA system is even more subjective than the evaluation of systems answering factoid or list questions (see Section 3.1.3 for a discussion of the evaluation metric

and (Voorhees, 2003b) for a thorough explanation of the problems with this approach to judging definition systems).

Most systems entered in the evaluation followed a similar architecture. Firstly, relevant sentences, i.e. those containing the target or an appropriately linked anaphor, were retrieved from the AQUAINT corpus. These sentences were then scored using a number of features, including the similarity of the target to known dictionary entries (such as WordNet glosses), biographies or definitions retrieved from the web and hand-crafted indicative patterns such as "TARGET is a kind of" or "TARGET known as". Methods borrowed from the summarization community were then applied to the ranked sentences in a bid to remove redundant sentences. Interested readers should consider consulting a selection of papers describing these systems, including Xu et al. (2003), Yang et al. (2003) and Echihabi et al. (2003). For a discussion of the evaluation metrics used and their reliability see Voorhees (2003b) and Section 3.1.3 of this thesis.

The relative immaturity of this area of QA research is well illustrated by that fact that the second best performing system evaluated as part of TREC 2003 was a simple baseline system entered by BBN (Xu et al., 2003). This system constructed its definitions by retrieving sentences containing the target and retaining the sentence if it did not overlap more than 70% with previously retrieved sentences, stopping when 4000 characters of text had been retrieved.

Chapter 3

Evaluating QA Systems

Evaluation can be a highly subjective task, especially when dealing with natural language systems. It is easier to evaluate tasks for which there is a more clearly defined answer (e.g. named entity recognition, which can be evaluated in terms of the proportion of entities correctly recognised), however, for most natural language tasks there is no single correct answer. For example, the method of evaluating information retrieval systems requires a text collection and a set of queries for which someone has manually searched the entire collection for all the relevant documents. Only then can the queries be used to make an evaluation of the system using recall and precision (defined later in this chapter). This is no easy task even for collections as small as the Cystic Fibrosis Database[1] (Shaw et al., 1991), which contains 1239 articles (approximately 5 megabytes of text) all of which would have to be compared with each query. Imagine trying to do the same for the AQUAINT collection used, in recent years for the TREC QA evaluations, which contains approximately 1,033,000 articles in 3 gigabytes of text (see Appendix A to appreciate how just how large some of these collections are). The expense of hiring humans to examine such large collections in order to generate gold standards for evaluation is also prohibitive, which further complicates the evaluation procedure.

The LUNAR system (Woods, 1973), designed to answer questions about the geology of moon rocks, is of historical interest as it was one of the first QA systems to be subject to user-evaluation (see Section 2.1). More recently the evaluation of QA systems has focused mainly on the QA track at TREC organised by the National Institute for Science and Technology (NIST).

3.1 End-to-End Evaluation

Most of the evaluations in this study will be concerned with the final output of a QA system and so a widely accepted evaluation metric is required to allows us to both evaluate the ideas detailed in this thesis and to compare our results with those reported by other QA researchers.

[1] `http://www.dcc.ufmg.br/irbook/cfc.html`

Most of the recent large scale QA evaluations have taken place as part of the TREC conferences and hence the evaluation metrics used have been extensively studied and could be of use in this study. What follows are definitions of numerous metrics for evaluating factoid, list and definition questions and an explanation of which metric will be used in the evaluations in Parts II and III of this thesis.

3.1.1 Factoid Questions

To evaluate QA systems we firstly need to define what constitutes an answer to a question. Clearly an answer has to be correct to be of any use, but this still leaves a lot of scope for different systems to present the same answer in different ways. Most of the systems we will look at in this thesis use unstructured text as their source of answers, and usually (but not always) return a short extract from a relevant document as an answer. The major difference between QA systems of this type is how long the returned answer is. While most real-world QA systems would probably return an answer in context (a phrase, sentence or paragraph which supports the answer) for the purposes of evaluation QA systems are usually expected to return *exact* answers. The reason for insisting on an exact answer is to determine how good systems are at pinpointing the exact answer. Isolating the exact answer would of course allow varying length sections to be taken from the document correctly centred on the answer, and it also opens up the possibility of using the exact answers along with text generation systems to provide answers which are not simply cut from a supporting document.

The main problem with getting systems to return exact answers is defining what makes an answer exact. For instance, most people would probably agree that *Mississippi*, *the Mississippi*, *Mississippi river* and *the Mississippi river* are all valid exact answers to the question *"Which river is known as the 'Big Muddy'?"*. It is, however, important to distinguish between extraneous material that is junk and extra material that further answers the question. For example, the question *"What year did the shuttle Challenger explode?"* has as an exact answer *1986*. Using a very strict definition of exact would mean that the more precise answer of *January 1986* would be deemed inexact as it provides more information than was requested. Clearly, *January* is extra material but it further answers the question and is not junk, many would consider this an ideal answer to the question. This is in fact the approach adopted throughout the work documented in this thesis; namely that an exact answer is anything that defines the answer and only the answer to a level of detail the same or better than was requested via the question.

Evaluation, irrespective of the metric being used, is carried out automatically. As most of the evaluations within this thesis make use of the TREC question sets and document collections (see Chapter 4) this can be easily achieved. After each TREC QA evaluation a set of regular expressions has been made available[2] which for each question match all those answers submitted to the evaluation which the human judges decreed were correct. It is also an easy task to generate from the judgment files (these contain the answers by

[2] Thanks to NIST and in recent years Ken Litkowski for producing these regular expressions.

each system to every question and a flag to mark them as correct, unsupported, inexact or incorrect) a list of documents from which at least one system has found a correct answer for each question. Using these two sources of data there are two ways in which we can determine if an answer is correct:

- **Strict:** An answer a to question q is considered correct if, and only if, it has previously been judged correct, i.e. it matches one of the regular expressions known to match correct answers to the question, and it is linked to a document d which is considered relevant.
- **Lenient:** An answer a to question q is considered correct if it has previously been judged correct, i.e. it matches one of the regular expressions known to match correct answers to the question.

Whilst neither approach produces an evaluation which is 100% correct they are at least consistent across different QA systems. Using the lenient method will always give higher results as unsupported answers (those were the linked document does not support the answer) will be considered correct. Throughout this thesis will we use the strict approach. Using the strict approach allows us to state that the results obtained are an accurate lower bound upon the results.

The original evaluation metric used in the QA tracks of TREC 8 and 9 was mean reciprocal rank (MRR). MRR provides a method for scoring systems which return multiple competing answers per question. Let Q be the question collection and r_i the rank of the first correct answer to question i or 0 if no correct answer is returned. MRR is then given by Equation 3.1:

$$MRR = \frac{\sum_{i=1}^{|Q|} 1/r_i}{|Q|} \tag{3.1}$$

As useful as MRR was as an evaluation metric for the early TREC QA evaluations it does have a number of drawbacks (Voorhees, 1999), the most important of which are that 1) systems are given no credit for retrieving multiple (different) correct answers and 2) as the task required each system to return at least one answer per question; no credit was given to systems for determining that they did not know or could not locate an appropriate answer to a question.

One improvement to MRR was to introduce a way of signifying that a system cannot determine the answer to a question (returning NIL as the answer) allowing systems to be credited for a correct NIL answer, if there is no known answer in the collection, in the same way as for any other correct answer (first introduced in TREC 10). This still does not cover the cases in which a system makes no attempt to answer a question (in this case most systems will return NIL possibly inflating their score if they do not attempt a question which happens to have no known answer in collection, see Appendix D).

As the QA TREC evaluation matured it became clear that the evaluation should move from evaluating systems over multiple answers per question to a single exact answer per

question. Clearly this change required a new evaluation metric. In TREC 11 confidence weighted score (CWS) was chosen as the new evaluation metric (Voorhees, 2002). Under this evaluation metric a system returns a single answer for each question. These answers are then sorted before evaluation so that the answer which the system has most confidence in is placed first. The last answer evaluated will therefore be the one the system has least confidence in. Given this ordering CWS is formally defined in Equation 3.2.

$$CWS = \frac{\sum_{i=1}^{|Q|} \text{number correct in first } i \text{ answers}/i}{|Q|} \tag{3.2}$$

CWS therefore rewards systems which can not only provide correct exact answers to questions but which can also recognise how likely an answer is to be correct and hence place it early in the sorted list of answers. The main issue with CWS is that it is difficult to get an intuitive understanding of the performance of a QA system given a CWS score as it does not relate directly to the number of questions the system was capable of answering.

As good as these evaluation metrics are another simpler metric is $accuracy$, the fraction of questions judged to have at least one correct answer in the first n answers to a question. Let $C_{D,q}$ be the correct answers for question q known to be contained in the document collection D and $F^S_{D,q,n}$ be the first n answers found by system S for question q from D then $accuracy$ is defined as:

$$accuracy^S(Q, D, n) = \frac{|\{q \in Q | F^S_{D,q,n} \cap C_{D,q} \neq \varnothing\}|}{|Q|} \tag{3.3}$$

Throughout the remainder of this thesis the evaluations of QA systems will be carried out using a strict form of the $accuracy$ measure with $a@n$ denoting $accuracy$ at rank n.

3.1.2 List Questions

List questions were originally a simple extension of the factoid questions. Instead of requiring a single exact answer list questions simply required a fixed sized, unordered set of answers, for example *"Name 22 cities that have a subway system"*. For these questions the score (for a single question) was simply the fraction of requested answers correctly returned. While this type of list question is relatively easy to evaluate the questions often seem artificial and may not represent the types of list questions real users would ask. For example it is unlikely that a users would want to know 22 cities which have a subway systems – they are more likely to ask if a given city has a subway system[3] or for a list of all cities which have a subway system.

From TREC 2003 the list questions have attempted to more accurately reflect real world users and no longer limit the target number of answer instances required. This change in question style clearly dictates a different evaluation metric which can cope with the unknown size of the answer set. Continuing with the notation introduced for the evaluation of factoid questions, let $C_{D,q}$ be the set of known answers to question q in the document

[3] An example of yes/no questions which are not covered by this thesis or the TREC evaluations.

collection D and $F^S_{D,q}$ be the response to question q by system S then we define the recall R of the answer set to be:

$$recall^S(D,q) = \frac{|F^S_{D,q} \cap C_{D,q}|}{|C_{D,q}|} \tag{3.4}$$

and the precision of the answer set to be:

$$precision^S(D,q) = \frac{|F^S_{D,q} \cap C_{D,q}|}{|F^S_{D,q}|} \tag{3.5}$$

The final score for a given question is a combination of precision and recall using the well known F measure, defined in Equation 3.6, giving an equal weighting (β equal to 1) to precision and recall.

$$F = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R} \tag{3.6}$$

Having now defined the metrics used to evaluate QA systems capable of answering list questions it is worth noting that very little will be said about these systems throughout the rest of this thesis as these questions have simply been treated as factoid questions in which all the answers (or all those above a given threshold) are returned as the answer set. While little will actually be said about list questions, note that all the ideas presented in Part II apply to answering list questions as well as factoid questions over which the evaluations are reported. See Appendices D and E for official TREC evaluations of list questions answered using the approaches developed for factoid questions in Part II.

3.1.3 Definition Questions

The answers to definition questions, such as *"What is aspirin?"* or *"Who is Aaron Copland?"* consist of an unordered set of text fragments (each fragment can be of any length although they usually range from single words to full sentences) which are judged against a set of *nuggets*. A nugget is a single atomic piece of information about the current target. For example there are seven nuggets that systems are expected to return for the question *"Who was Alexander Hamilton?"*:

1. Secretary of the US Treasury
2. killed in duel with Arron Burr
3. charged with financial corruption
4. congress refused to impeach
5. confessed to adultery
6. leader of the federalist party
7. named chief of staff by Washington

Nuggets come in two flavours; vital and acceptable. A good answer to a definition question should contain all the vital nuggets but will not be penalised for containing any or all of the acceptable nuggets. In the above example the first three nuggets are considered vital while the remaining four are acceptable nuggets of information for a system to return. Answers to definition questions are formally evaluated using nugget recall and precision.

Given a set V of vital nuggets and the set v of vital nuggets returned by a system then nugget recall, NR, is simply the proportion of the known vital nuggets returned by the system:

$$NR = \frac{|v|}{|V|} \tag{3.7}$$

To define nugget precision, NP, we need firstly to define two other measures; $length$ and $allowance$. The $length$ of a definition is the total number of non-whitespace characters which make up the definition. Each nugget (both vital, v, and acceptable, a) contained in the definition gives an $allowance$ of 100 non-whitespace characters in which the definition should be contained and is formally defined as:

$$allowance = 100 \times (|v| + |a|) \tag{3.8}$$

Nugget precision, NP, is then defined so as to determine how close in length the system response is to the $allowance$, i.e. the $allowance$ calculates the maximum size of a definition before it is penalised for verbosity. The formal definition of NP is given in Equation 3.9.

$$NP = \begin{cases} 1 & \text{if } length < allowance, \\ 1 - \frac{length - allowance}{length} & \text{otherwise.} \end{cases} \tag{3.9}$$

Nugget precision and recall are combined using F measure as defined by Equation 3.6 (with NP and NR replacing P and R respectively).

The main problem with the evaluation metric is that the ranking of different systems can change quite dramatically given a change in β in the F measure equation. The TREC 2003 evaluations used a β value of 5 based on the correlation between results obtained using this configuration and a *holistic* evaluation carried out as part of a pilot evaluation held in 2002 (Voorhees, 2003a). Figure 3.1 shows the performance of the top 5 TREC 2003 systems (evaluated with β equal to 5) and a baseline system (Xu et al., 2003) for β values 1 to 5 (Voorhees, 2003b).

As you can clearly see from Figure 3.1 the value of β not only has an affect on the actual score assigned to a given system but more importantly can dramatically alter the relative performance of different systems. For instance the sentence baseline performs exceptionally well when β is set to 5 but quite poorly at lower values (when the best run from all participating institutions are considered then it is ranked 11^{th} when β is 1 giving an F score in which recall and precision are equally weighted).

Even if it was possible to determine the correct β value based on extensive comparison between human and computer evaluations an issue still remains – it is exceptionally time

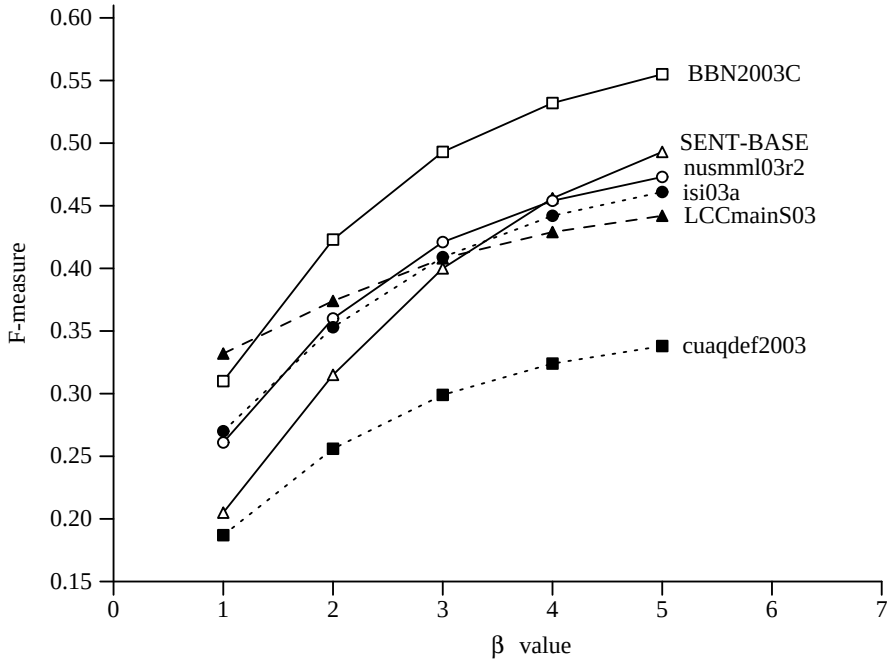

Figure 3.1: Effects of varying β on the evaluation of definition systems.

consuming to evaluate a definition run and as yet, unlike for factoid (and to some extent) list questions, no accepted automatic evaluation method exists.

Recently published work (Xu et al., 2004) suggests that it may be possible to use the Rouge evaluation method (Lin and Hovy, 2003), originally developed for the evaluation of computer generated summaries, to automatically evaluate answers to definition questions. Given a reference answer, R, and a system response, S, the Rouge score for evaluating definition questions can be derived from the work by Lin and Hovy (2003) to give Equation 3.10 in which $count_{match}(R, S, n)$ is the number of shared n-grams between R and S, and $count(R, n)$ is the number of n-grams in R, with Xu et al. (2004) suggesting 3 as an appropriate value for m.

$$Rouge(R, S, m) = \sqrt[m]{\prod_{n=1}^{m} \frac{count_{match}(R, S, n)}{count(R, n)}} \tag{3.10}$$

The main outstanding issue hindering a wide adoption of Rouge is the requirement of a reference answer. The current assessor lists issued by NIST whilst containing the answers tend to be terse notes rather than well formed phrases. A reference answer of well formed phrases was manually formed by interpreting each nugget with reference to a selection of relevant documents to provide background context. Until a widely accepted set of reference answers is made available, evaluations using Rouge will be open to interpretation and not directly comparable with other evaluations.

While the idea of an automatic evaluation method is certainly appealing the evaluations in Part III of this thesis will continue to use the TREC evaluation metric, that is an F score with a β value of 5, to allow easy comparison with currently available data from TREC

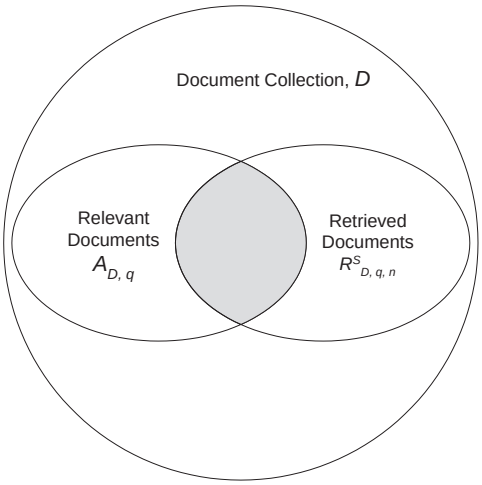

Figure 3.2: Sections of a document collection as used for IR evaluation.

2003. Note that TREC 2004 will use a β value of 3 and there is currently no consensus within the community on the correct value of β so due consideration should be given to evaluations using β values ranging from 1 to 5.

3.2 Evaluating IR Systems for QA

Many QA systems follow a basic three part architecture (see Chapter 5) incorporating an information retrieval system which selects a small subset of documents for later stages of the system to process in detail to extract possible answers. Given the importance of such a component in the overall performance of a QA system the IR engine should also be evaluated in some detail. If the IR component does not return any answer bearing documents then no amount of further processing will produce a correct answer.

3.2.1 Traditional Metrics

The standard evaluation measures for IR systems are precision and recall. Let D be the document (or passage collection), $A_{D,q}$ the subset of D which contains relevant documents for a query q and $R^S_{D,q,n}$ be the n top-ranked documents (or passages) in D retrieved by an IR system S. Figure 3.2 illustrates the sets involved. In the context of IR for QA a relevant document is one which contains a correct answer to the question with enough context to support the answer.

The $recall$ of an IR system S at rank n for a query q is the fraction of the relevant documents $A_{D,q}$ which have been retrieved:

$$recall^S(D, q, n) = \frac{|R^S_{D,q,n} \cap A_{D,q}|}{|A_{D,q}|} \tag{3.11}$$

The $precision$ of an IR system S at rank n for a query q is the fraction of the retrieved

documents $R^S_{D,q,n}$ that are relevant:

$$precision^S(D,q,n) = \frac{|R^S_{D,q,n} \cap A_{D,q}|}{|R^S_{D,q,n}|} \tag{3.12}$$

Clearly given a set of queries Q average $recall$ and $precision$ values can be calculated to give a more representative evaluation of a specific IR system.

Unfortunately these evaluation metrics although well founded and used throughout the IR community suffer from two problems when used in conjunction with the large document collections utilized by QA systems, namely determining the set of relevant documents within a collection for a given query, $A_{D,q}$. The only accurate way to determine which documents are relevant to a query is to read every single document in the collection and determine its relevance. Clearly given the size of the collections over which QA systems are being operated this is not a feasible proposition (see Appendix A to appreciate how infeasible this really is). Fortunately *pooling* has previously been shown to be both a reliable and stable method of evaluating IR systems (Buckley and Voorhees, 2004). In pooling a list of known relevant documents is generated from a sampling of the responses being evaluated. In TREC, for instance, for a single query the first 100 responses returned by each system being evaluated is checked for relevance and these pooled responses are then used to evaluate the 1000 documents retrieved by each system. Whilst pooling allows stable discrimination between systems recall tends to be overestimated as only 50%-70% of relevant documents are located (Zobel, 1998).

A more important problem with evaluating IR systems for use in QA systems relates directly to the use of the traditional measures of precision and recall.

Consider two IR systems S_1 and S_2 being used to find relevant documents to 100 questions all of which are known to have 100 relevant documents within the collection being used. If S_1 returns 100 relevant documents for question 1 and no relevant documents for the other 99 questions then it will have average recall and precision of 0.01. If on the other hand S_2 returns a single correct answer for each of the 100 questions then it will also have average recall and precision of 0.01. Therefore, any QA system using S_1 as its IR component will be able to return a correct answer for at most 1 question, whereas when using S_2 it would be able to return a correct answer for any of the questions (just because a relevant document is found does not automatically mean the QA system will be able to identify and extract a correct answer). From a QA perspective system S_2 is substantially better than S_1 although the traditional measures of precision and recall are unable to differentiate between the two systems.

3.2.2 Alternative Measures

In a naïve attempt to provide replacement evaluation methods more suited to use over large collections (such as the Internet and for evaluating web search engines in particular) we define two new measures $position$ and $answer$ (Greenwood, 2001). The $position$ is simply the average position at which the first relevant document appears within the

first ten results and $answer$ is defined as the percentage of queries for which at least one relevant document appears within the top ten documents. Both measures were restricted to considering only the first ten answers as they were designed to evaluate the output of web search engines and most people like to see an answer to a query on the first page of results with search engines displaying ten results per page. Clearly both measures are overly simplistic although $answer$ does allow us to distinguish between the example systems S_1 and S_2, introduced earlier, which would have answer values of 1% and 100% respectively.

In response to the issues surrounding the use of traditional IR evaluation metrics in the context of QA Roberts and Gaizauskas (2004) developed two new metrics specifically for evaluating IR systems which are used as part of larger QA systems: $coverage$ and $redundancy$. Continuing with the same notation as before we also introduce Q to denote a set of questions.

The $coverage$ of a retrieval system S for a question set Q and document collection D at rank n is the fraction of the questions for which at least one relevant document is found within the top n documents:

$$coverage^S(Q, D, n) = \frac{|\{q \in Q | R^S_{D,q,n} \cap A_{D,q} \neq \varnothing\}|}{|Q|} \tag{3.13}$$

The $answer\ redundancy$ of a retrieval system S for a question set Q and document collection D at rank n is the average number of relevant documents found within the top n documents:

$$answer\ redundancy^S(Q, D, n) = \frac{\sum_{q \in Q} |R^S_{D,q,n} \cap A_{D,q}|}{|Q|} \tag{3.14}$$

From the definition we can see that $coverage$ is a generalisation of the naïve method $answer$ allowing us to compare systems at any given n and also that $coverage$ is similar to the $accuracy$ measure, used for evaluating the end-to-end performance (see Section 3.1), but concerned with determining which documents contain correct answers to a question rather than which system responses are correct answers to a question.

The main advantage of the metrics $coverage$ and $answer\ redundancy$ are that they do not require us to know the complete set of relevant documents in D for query q, rather we need only know which of the retrieved documents are relevant to the query, $R^S_{D,q,n} \cap A_{D,q}$ and fortunately there is an easy way to approximate this. In the previous section we described an automatic way of evaluating QA systems using resources distributed via TREC. These same resources can also be used to provide a strict and lenient evaluation procedure for measuring the performance of competing IR components:

- **Strict:** A document d is considered relevant (i.e. $d \in A_{D,q}$) if, and only if, it has been judged to contain an answer to the question q and one of the regular expressions known to match correct answers to q matches the content of d (both tests are required to allow passage retrieval systems to be correctly evaluated).

- **Lenient:** A document d is considered relevant if one of the regular expressions for question q matches the content of d.

Just as with the evaluation of the end-to-end performance of a QA system both measures of relevance can be used to compute coverage or redundancy. Neither approach will produce values which are 100% correct although they will be consistent over different IR approaches.

Increasing the coverage for a document set by simply retrieving more documents is of course unlikely to result in better end-to-end performance of a QA system because this will also result in an increased noise level within the document set (see the next section for a definition of noise). What is useful, however, is to compare the coverage of two different IR approaches over the same volume of text. As most of the experiments in this thesis retrieve documents (or passages) of a similar size then a sensible approach would be to compare the difference in coverage (or redundancy) at a given retrieval rank. See Saggion et al. (2004) for further examples of the volume of text being an important aspect of the evaluation and comparison of differing IR systems and approaches.

Throughout the remainder of this thesis we will quote the evaluation results calculated using the strict approach. Using the strict approach allows us to state that the results obtained are an accurate lower bound upon the results. Coverage at rank n will be represented as $c@n$ with redundancy being $r@n$.

3.2.3 Noisy Documents

An increase in coverage while more significant than an increase in the traditional IR metrics of precision and recall does not guarantee an improvement in the performance of a full QA system.

Many QA systems determine, during question analysis (see Chapter 6), the semantic type of answers to a question and use this information to check or refine the answer extraction phase (see Chapter 8). While novel approaches to IR may increase the coverage of the retrieved documents they may also increase the number of entities of the answer's expected semantic type within the set of retrieved documents, i.e. the search space over which an answer extraction component operates may increase. If this increase in the search space leads to more instances of the correct answer then the answer extraction components may be able to extract the correct answer without a problem. It is possible, however, that the percentage of entities of the expected answer type which are correct answers to a question will decrease, i.e. there will be a greater level of noise which may distract the answer extraction components away from the correct answer or answers.

Let D be the set of relevant documents retrieved by an IR system S, for a question q taken from a set of questions Q, t be the expected answer type[4] of question q and $I_{D,t,q}$ be the

[4] In this instance 'answer type' refers not just to standard semantic types such as Person, Date and Lo-

set of unique instances of type t found in the documents D for question q while $A_{D,t,q}$ is the set of unique answer instances of type t found in the documents D which correctly answer question q.

The $noise$ of an IR system S and question set Q is the fraction of the answer instances $I_{D,t,q}$ judged incorrect and is formally defined as:

$$noise^S(Q) \equiv \frac{\sum_{q \in Q} \frac{|I_{D,t,q} - A_{D,t,q}|}{|I_{D,t,q}|}}{|Q|} \tag{3.15}$$

So as well as determining the fraction of questions which could be answered using coverage we can calculate $noise$ in an effort to determine how difficult it is to determine which the correct answer is out of the set of entities of the correct type. In the results detailed in this thesis we will refer to $noise$ at rank n as $n@n$.

3.3 Are These Two Results Truly Different?

When comparing different approaches to a problem it is important not just to show that one system gives better results than another but whether or not the differences between the approaches are significant and not due to random chance. To determine if two results really are different, experimental results in this thesis are compared using the paired t test (Kirkwood, 1988).

Assuming that comparisons between two systems X_1 and X_2 are carried out by evaluating the systems over the same set of n independent questions, let Y_i be the performance difference between systems X_1 and X_2 for the i-th question:

$$Y_i = X_{1,i} - X_{2,i} \tag{3.16}$$

When evaluating binary results of system X_s (i.e. IR and factoid questions in which either the answer is present or not) on question i the performance $X_{s,i}$ is defined as:

$$X_{s,i} = \begin{cases} 1 & \text{if } X_s \text{ correctly answered question } i \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

For results with absolute values, such as the F measure scores assigned to the answers for list and definition questions, $X_{s,i}$ is simply the score assigned to the answer of question i produced by system s. Given the performance difference, Y_i, we can determine the sample mean, $\bar{Y}$ over n sample questions using Equation 3.18:

$$\bar{Y} = \frac{\sum_{i=1}^{n} Y_i}{n} \tag{3.18}$$

cation but to much more complex concepts. For instance the answer type of the question *"How did Mahatma Gandhi die?"* would contain not only nouns such as *heart attack* and *cancer* but more complex concepts such as *'murdered by X'*. Although it is not always possible for systems to recognise all instances of a complex answer type, this is not important for the calculation of the noise level within a document set, as the noise level is based solely upon the correct and incorrect answers instances the system is able to recognise.

Using both the performance difference Y_i and sample mean $\bar{Y}$, the sample variance, s^2, is defined as:

$$s^2 = \frac{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}{n-1} \tag{3.19}$$

The paired t test is then defined as:

$$\text{Paired } t = \frac{\bar{Y}}{\sqrt{s^2/n}} \tag{3.20}$$

The larger the value of t, the less likely it is that the difference between the two systems under evaluation is due purely to chance (note that the sign of t is ignored). On a reasonably sized sample of questions (at least 30) a value of t over 1.65 signifies that such a difference being due to chance is less than 5% with a t value over 2.33 signifying a less than 1% chance. For smaller samples the percentage point values increase showing that is more difficult to prove a statistical significance with only a few sample questions[5](for example, with only 10 questions t needs to be greater than 1.83 or 2.82 for 5% and 1% respectively).

Throughout the remainder of this thesis improvements in the results which are significantly different with 99% confidence (difference due to chance is less than 1%) are signaled by ▲ while △ signifies only 95% confidence. Similar meanings are attached to ▼ and ▽.

[5] See `http://www.statsoft.com/textbook/sttable.html` for t cut-off values.

Chapter 4

Experimental Setup

The main aim of the research detailed in this thesis is to investigate a number of novel ideas and techniques which, it is hoped, will contribute to the ever expanding field of question answering. To be of real use to the QA research community, evaluations of new techniques should be easily comparable with those of other researchers in the field. As many researchers submit their QA systems for independent evaluation as part of the TREC conferences, they tend to publish results of their own in-house evaluations over the same questions and document collections used for the TREC evaluations. For this reason these will be the main source of data used to evaluate different techniques throughout this thesis.

In the same way that Chapter 3 detailed commonly used evaluation metrics which allow systems to be evaluated and the results compared, this chapter describes in some detail, not only the data collections used throughout this thesis to evaluate the techniques presented but also further details of some of the more subtle points of document collections and question sets which can have an effect on the evaluations of QA techniques.

4.1 Document Collections

Throughout this thesis most of the evaluations make use of a document collection known as AQUAINT which has been used for the TREC QA evaluations since 2002. The AQUAINT Corpus of English News Text[1] consists of approximately 1,033,000 documents in 3 gigabytes of text covering the Associated Press Worldstream News Service from 1998 to 2000, the New York Times news service from 1998 to 2000, and the English portion of the Xinhua News Service (People's Republic of China) from 1996 to 2000. By using the AQUAINT collection (along with the questions used in the TREC evaluations detailed in the following section) we can reliably compare performance across not only our own competing ideas and techniques but also those from other research groups knowing that the systems are working from the same initial knowledge source.

[1] Linguistic Data Consortium (`http://www.ldc.upenn.edu`) catalogue number LDC2002T31.

While the AQUAINT collection contains both a vast and varied amount of information, certain techniques require the use of even larger and more varied collections. For example the surface matching text patterns, detailed in Section 8.2, require a collection to not only contain the correct answers to many questions of the same type but to also contain many instances of the answer to each question. For this kind of technique we make use of the web as a document collection (usually accessed via Google).

Irrespective of the document collection being used, each and every document is assumed to contain only true factual information. In essence this means that even if a QA system were to return an incorrect answer to a question, as long as the document from which it was taken supports the answer being correct then it is evaluated as such. As well as covering erroneous information this also applies to the problem of temporally dependent answers. For example consider the question *"Who is the governor of Colorado?"* which was asked as part of the 2002 TREC evaluation. In 2002 the governor of Colorado was Bill Owens and an example supporting document from AQUAINT makes this clear (XIE19990412.0037):

> *In Denver, Zhu met Colorado Governor Bill Owens, Denver Mayor Wellington Webb and representatives from the local Chinese community.*

As the AQUAINT collection contains documents covering the period 1996 to 2000 there was of course no guarantee that Bill Owens would still be the governor in the summer of 2002, especially as the supporting document is a news article dating from the 12th of April 1999. A more serious issue can be seen by examining the following excerpt from another document in the AQUAINT collection (NYT19981113.0016)

> *Boston's place on the list is no surprise because Grossman is from Massachusetts, while Denver's selection may have something to do with the fact that DNC co-chairman Roy Romer is the governor of Colorado.*

This document clearly states that the governor of Colorado is in fact Roy Romer. This can be explained by the fact that the document pre-dates Bill Owens' term as governor (he was officially sworn into the office in January 1999), being dated the 13th of November 1998. Both documents clearly state, however, that different people are the governor of Colorado (neither document says ex-governor or in any way suggest that either is no longer the governor of Colorado) and hence, given the strict evaluation metric detailed in Chapter 3, both Roy Romer and Bill Owens are considered correct answers to the question.

Clearly the above example could be solved simply by assuming that the answer in the most recent article should be considered correct, although this is not without its own problems (historical articles often contain text such as *"...as Governor of Colorado Roy Romer helped to establish..."* which alone is not enough to determine if this is present or past tense and so the article date may be misleading). One possible solution would be to improve the current range of QA systems to incorporate a more complete world

knowledge representation, so that documents could be correctly understood and related to each other. Humans when answering questions rely on world knowledge to enable them to answer questions and deal with just such conflicting data (Deglin and Kinsbourne, 1996; McManus, 2002) and so it is likely that advanced QA systems will also require some level of world knowledge.

What should be clear is that no matter how good a QA systems is if it relies solely on the documents from which it can extract answers then there is always a possibility of the answers being *factually* wrong due to some form of falsehood in the text (erroneous or out-of-date information being the most likely) although the answer will in fact be evaluated as correct. Only world knowledge will overcome this difficulty, although it is not clear how best to integrate large scale world knowledge with current state of the art QA systems.

4.2 Question Sets

Just as the choice of document collection was focused around allowing experimental results to be comparable to those of other researchers so is the choice of questions over which the approaches will be evaluated. There are, as of autumn 2004, three accepted question sets associated with the AQUAINT collection. These questions sets are those which were used for the 2002, 2003 and 2004 TREC QA evaluations.

The format of the questions used in the 2004 evaluation is, however, different from that used in previous years and from the style of questions considered in this thesis. In the 2004 evaluation a number of targets were listed along with a small set of related questions as a scenario. For example target 3 was *"Hale Bopp comet"* for which the questions were:

> Q3.1 (Factoid): *When was the comet discovered?*
> Q3.2 (Factoid): *How often does it approach earth?*
> Q3.3 (List): *In what countries was the comet visible on its last return?*
> Q3.4 (Other): This translates to *'tell me anything else which is interesting and relevant but for which I did not know enough about the target to ask directly'*.

This means that some of the questions cannot be asked in isolation from the target without the use of appropriate coreference resolution to place the target within the question (e.g. Q3.2 does not actually contain the target). These questions will not be considered in the evaluations of techniques documented in this thesis, although official TREC 2004 evaluations of the main approaches from Parts II and III can be found in Appendix E.

The majority of the evaluations detailed in Part II of this thesis, which deal with answering factoid questions, will use the questions from the TREC 2002 QA evaluation. This question set contains 444 questions which have at least one known correct answer within the AQUAINT collection and for which there is an accepted regular expression which matches against instances of the correct answer (both of these requirements are necessary to perform strict evaluations as described in Chapter 3). In cases (such as those in Section 6.2)

where evaluating an approach requires a number of questions of a similar type use was also made of the questions prepared for the TREC 2003 QA evaluation.

Whilst participating in both TREC 2002 and 2003 involved performing failure analysis and hence checking the performance of a number of QA components (not all of which are covered by this thesis) over the associated questions, no attempt was made to fix specific issues which arose from this failure analysis. That is, the questions used in the evaluations in this thesis were, as far as possible, used as a blind evaluation of the approaches. As all the research contained in this thesis was completed before the TREC 2004 QA evaluation this official evaluation represents an entirely blind and independent evaluation of the work and is discussed in some detail in Appendix E.

The approaches to answering factoid questions detailed in Part II were developed using training questions gathered from a number of sources. Firstly the questions used in the TREC QA evaluations before 2002 were used as were a collection of 10,246 Pub Quiz style questions (see Appendix 1 of Sargaison (2003)) collated from the publicly available websites of The Calveley Arms[2], The Pear Tree[3], The Midland Pub[4] and Quiz-Zone[5]. This collection (unlike those assembled for TREC) has an unmistakably English bias in that the questions make frequent reference to minor English celebrities, television programmes and items of only national and not international interest. A brief attempt has been made to remove temporal reference by filtering out those questions containing terms such as today, last week, this month, etc., leaving questions of a similar style to those used in the TREC QA evaluations.

The evaluations of the approaches to answering definitional questions, described throughout Part III of this thesis, are performed using the 50 definition questions assembled for the TREC 2003 QA evaluation. Unfortunately there are no earlier large scale sets of definition questions which could be used for development so as to leave the TREC 2003 questions as a blind test set. For this reason the questions were also used during development but only as a method of evaluating the performance of the approach and not as a guide to the direction of the work. As with the factoid approaches an independent evaluation of the work on answering definition questions was carried out as part of the 2004 TREC QA evaluation details of which can be found in Appendix E.

Whilst this section has detailed the main question sets used during the development and evaluation of the techniques presented in this thesis, other questions have, on some occasions, been required. Where an approach requires extra sets of questions and answers their source is documented along with the approach.

[2] `http://fp.geoffwilliams.plus.com/pubquizzesarchive.htm`

[3] `http://www.stan.kurowski.btinternet.co.uk/quizzes.html`

[4] `http://www.stubaby.plus.com/quiz/quiz.htm`

[5] `http://www.quiz-zone.co.uk/`

Part II

How To Answer Questions

Chapter 5

A Framework for QA

The majority of current question answering systems designed to answer factoid or list questions (and to some extent definition systems, see Part III) consist of three distinct components: question analysis, document or passage retrieval and answer extraction. While these three components can and often are sub-divided into lower level operations, such as document collection pre-processing and candidate document analysis (Hirschman and Gaizauskas, 2001), a three component architecture describes the approach taken to building QA systems both in this thesis and in the wider literature. The architecture of a generic three component QA system can be seen in Figure 5.1.

It should be noted that while the three components address completely separate aspects of question answering it is often difficult to know where to place the boundary of each individual component. For example the question analysis component is usually responsible for generating an IR query from the natural language question which can then be used by the document retrieval component to select a subset of the available documents. If, however, an approach to document retrieval requires some form of iterative process to select *good quality* documents which involves modifying the IR query, then it is difficult to decide if the modification should be classed as part of the question analysis or document retrieval process. While this is not an issue when developing a QA system it can present a challenge when attempting to document the different approaches to QA. Throughout this

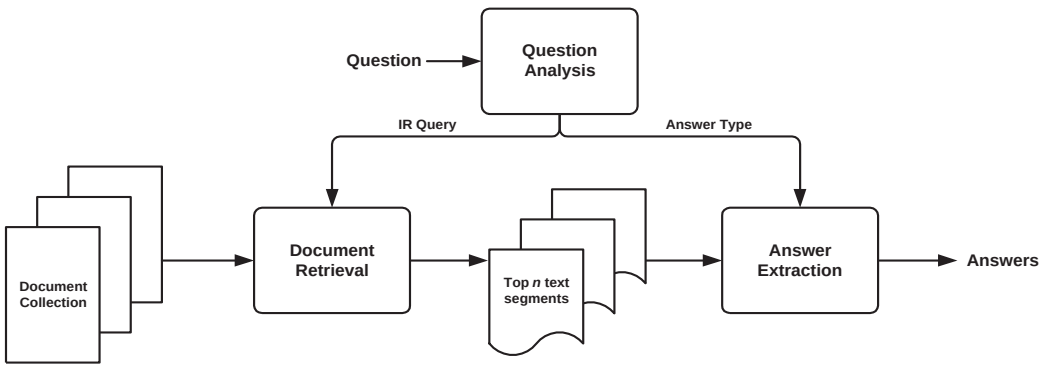

Figure 5.1: A framework for question answering.

thesis approaches which seem to cross component boundaries will be described where it is most appropriate.

This chapter briefly outlines the roles played by the three components in answering factoid questions. Each component is described by way of examples of a number of approaches that have been reported in the literature for solving the issues and challenges related to it. The following three chapters then give details of specific approaches that have been developed and evaluated as part of this thesis.

5.1 Question Analysis

The first stage of processing in any QA system is question analysis. In general the input to this first stage of processing is a natural language question. However, it should be noted that some systems simplify the analysis component either by using only a subset of natural language both in terms of syntax and vocabulary or by some form of restricted input method. For example many natural language interfaces to databases have some restrictions on the language that can be used and some question answering systems, such as SHAPAQA[1] (Bucholz and Daelemans, 2001), require the information need to be specified explicitly through a form style interface.

Another issue that question analysis components have to deal with is that of context. If each question is entirely self-contained and independent from any other question then the system can process it as such. If, however, the question is part of a longer sequence of questions (often referred to as context questions or a question scenario) then knowledge such as the answer or scope of previous questions may need to be considered, as may anaphora in the question which will require processing to find the correct antecedent before the question can be answered. This is the direction QA research seems to be taking with the questions in the TREC 2004 QA evaluation being divided into sets of questions about a given target entity (Voorhees, 2004).

The role of the question analysis component is to produce as output a representation of the question which can be used by the rest of the system to determine the answer to a question. Often this actually involves producing multiple output representations for different parts of the system. For example, question analysis components may supply the document retrieval component with a correctly formed IR query generated from the question whilst passing the answer extraction component some form of semantic representation of the expected answer. A number of papers have been published which deal with both aspects of question analysis and what follows is a brief tour through the available literature.

[1] `http://ilk.kub.nl/shapaqa/`

5.1.1 Producing an IR Query

Most if not all question analysis components create an IR query by starting with a bag-of-words approach – that is all the words in the question ignoring case and order are used to construct a query. The differences between systems lie in how the individual words are then treated to determine the final IR query.

One common processing step is to stem all the question words, usually via the Porter stemmer (Porter, 1980), to remove morphological variation (note that this implies that the documents being searched were also subjected to stemming before being indexed). This may in fact be part of the IR engine or may be performed separately by the question analysis component. As an example the Porter stemmer reduces the words *walk*, *walks*, *walked* and *walking* to the word *walk*, which means that searching using any of the four variants of *walk* will match against documents containing any of the four variants. While this seems ideal, as documents containing any of the four variants are likely to be relevant to a question containing the word *walking*, it is not without its own problems. For instance *heroine*, the main good female character in a work of fiction, stems to *heroin*, the addictive narcotic, which is itself unchanged by the stemmer. This leads to the unwanted situation in which searches for documents about *heroines* also return documents about *heroin* which are highly unlikely to contain the correct answer. Bilotti et al. (2004) go further and show that in some cases performing stemming decreases the overall IR performance compared with a simple bag-of-words approach. They show that both recall and the ranking of relevant documents is negatively affected by stemming (they evaluate the ranking of relevant documents using a weighted recall approach referred to as total document reciprocal rank). As an alternative to stemming they suggest that systems should attack the problem of morphological variation by constructing a query containing the different morphological variants of the question terms. For example the question *"What lays blue eggs"* would, under the three different approaches, be converted to:

Bag-of-Words: blue $\wedge$ eggs $\wedge$ lays

Stemming: blue $\wedge$ egg $\wedge$ lai

Morphological Expansion: blue $\wedge$ (eggs $\vee$ egg) $\wedge$ (lays $\vee$ laying $\vee$ lay $\vee$ laid)

While removing the issues associated with different words stemming to the same root form it should be clear that generating queries containing morphological expansions will both increase the size of the index into the document collection (as all morphological variants will require their own separate entry in the index) and require more processing to generate the lists of alternatives. One further issue with morphological expansion reported by Bilotti et al. (2004) is that while more relevant documents are retrieved this tends to be at the expense of moving relevant documents further down the rankings produced by the IR engine, i.e. the approach seems to have a greater effect on irrelevant documents than relevant ones which could have a detrimental effect on the performance of some answer extraction components.

Whichever method of dealing with morphological variation is adopted, other issues with

query formulation still exist. Standard IR queries are usually targets about which the system should locate relevant documents, meaning that most of the required information is present in the query. In question answering the situation is a little different as we are not attempting to locate documents about a specific target but rather some aspect or attribute of the target which may or may not be the main focus of the documents which contain answers to the question. It may also be the case that as the focus of the search is not explicitly mentioned in the question there may in fact be little in common between the question and the answer bearing documents. For example the question *"Where was Hans Christian Anderson born?"* makes no explicit mention of the answer (obviously!) but also makes no reference to anything else which could be the main focus of relevant documents. For instance it is quite possible that an article about Denmark could quite easily contain the text *"... the birthplace of Hans Christian Anderson and home to..."* which while answering the question has little in common with it, in fact no words other than the name are present in both. On the other hand a document about Hans Christian Anderson's stories could mention his name quite frequently and hence be highly relevant as it will contain many of the query terms, without ever mentioning where he was born.

A number of approaches to solving this problem have been tried including synonym expansion (Harabagiu et al., 2000; Hovy et al., 2000) and expected answer type expansion (Monz, 2003). While both approaches show promise they often lead to complicated queries which have to be iteratively relaxed before a reasonable number of documents can be retrieved. They also introduce other problems. For example, synonym expansion requires accurate word sense disambiguation to be performed over the question before the correct senses can be determined and expansion can take place.

Clearly the issue of performing IR within the context of QA is a difficult problem, one worthy of future work. Unfortunately it has been largely ignored by the QA community with many researchers relying instead on off-the-shelf IR engines.

5.1.2 Determining the Expected Answer Type and Constraints

As previously mentioned the question analysis component, as well as generating a query to be passed to the document retrieval component, also determines possible constraints on the expected answer which can be used by answer extraction components to constrain the search space. These constraints are usually semantic in nature. For example, any correct answer to the question *"Where was Hans Christian Anderson born?"* will be some form of location.

A naïve approach to determining expected answer type is to assign an answer type based upon the main question word: *when* signifying a date, *where* a location and *who* usually a person. Unfortunately while this works for simple examples, questions such as *"What university did Thomas Jefferson found?"* require more detailed processing to determine that the answer type is University (a sub-type of Organization).

One approach taken by many researchers is to construct a hierarchy of answer types and

then induce or construct a classifier that given an unseen question will assign it a type from the hierarchy. For example, Li and Roth (2002) define a two layer hierarchy consisting of 6 coarse classes and 50 fine classes (see Table 6.1) with each coarse class containing a non-overlapping set of fine classes, while Hovy et al. (2000) developed a more complex hierarchy, consisting of 94 nodes, by manually analysing 17,384 questions.

Given a question hierarchy, systems require a method of assigning a specific class to unseen questions. Numerous machine learning approaches including naïve Bayes, decision trees and support vector machines have been used with results over the fine grained classification task ranging between 68% and 85% (Li and Roth, 2002; Zhang and Lee, 2003). Another approach to determining the answer type is to modify a semantic parser to produce output which contains the question classification, an approach taken by among others the Webclopedia (Hovy et al., 2000) and QA-LaSIE (Greenwood et al., 2002) systems.

Once the expected answer type has been determined the remaining task of the question analysis component is to determine any remaining constraints on the answer which may allow answer extraction components to narrow their search space. Again numerous approaches have been reported from simple keyword extraction for windowing methods to full parsing to discover relationships such as subject-verb-object relationships. For example QA-LaSIE (Greenwood et al., 2002) parses the question *"Who wrote Hamlet?"* to produce the quasi-logical form (QLF) representation (Gaizauskas et al., 2005b; Gaizauskas et al., 2005c):

```
qvar(e1), qatr(e1,name), person(e1), lsubj(e2,e1),
write(e2), time(e2,past), aspect(e2,simple),
voice(e2,active), lobj(e2,e3), name(e3,'Hamlet')
```

From this QLF representation of the question the system can determine not only that the expected answer type is Person (`qvar` represents the entity being sought, `e1`, which according to the QLF is of type person) but also that the person we are looking for as the answer may well occur in the subject position of the verb 'to write' with the object of the verb being *Hamlet*.

Of course alternative methods of determining constraints are also feasible. For instance users will expect the answer to the question *"Who is Tom Cruise married to?"* to be a person. We can determine a further constraint that the answer be female as we know that marriages *usually* involve a man and a woman and so, as the question contains a male name, we can assume that the answer will be a woman and vice-versa.

5.2 Document Retrieval

Although, as has already been mentioned, the question analysis component is usually responsible for constructing an appropriate IR query this does not mean that the document retrieval component consists solely of using the query to pass on relevant documents to the answer extraction component. In fact the document retrieval component can be

quite complex involving numerous strategies for retrieving both documents and associated data as well as determining the amount and structure of the text to be passed to the final component in the system pipeline.

Full syntactic and semantic parsing can be a time consuming process but one which many QA systems rely on for answer extraction. The time required to produce full syntactic or semantic representations often limits their use within real-time question answering. One solution to this, at least for QA over closed collections, would be to pre-parse the whole collection storing the resulting parse trees and semantic representations. ExtrAns (Mollá Aliod et al., 1998) is one system which does just this – deriving and storing logical representations of all documents in the collection in advance of any question answering. Of course other less intensive and time consuming processing can also be carried out in advance. In fact there is no fundamental reason why all question independent processing (tokenization, POS tagging, named entity detection ...) cannot be carried out in advance, as long as the time to retrieve the data is not longer than that taken to generate it as this would defeat the main use of pre-processing – faster question answering.

The main role of the document retrieval stage is, however, to retrieve a subset of the entire collection which will be processed in detail by the answer extraction component. The main issues arising at this point are which IR paradigm to adopt (ranked or boolean) and the volume and structure of text to retrieve.

Many QA systems make use of ranked IR engines, such as Okapi (Robertson and Walker, 1999), although a number of researchers have suggested that boolean retrieval is better suited to the problems of question answering (Moldovan et al., 1999; Saggion et al., 2004). Query formulation for boolean IR approaches is inherently more complex as due care and consideration has to be given to how to rank or restrict an unordered list of relevant documents, something performed automatically when using a ranked IR engine. The main advantage of boolean systems is that their behaviour is easier to inspect as the matching process is inherently transparent with little or no interaction between separate search terms.

Logic would suggest that the document retrieval component should return a large number of documents for each question so as to ensure that at least one relevant document is retrieved, especially as reported results (Hovy et al., 2000) show that even when considering the top 1000 text segments no relevant documents are retrieved for 8% of the questions (these results were obtained when using the MG IR engine (Witten et al., 1999a)). Unfortunately experience (see Section 7.1) and published results (Gaizauskas et al., 2003) shows that while increasing the amount of text does indeed increase the answer coverage it can also decrease the accuracy (as defined by Equation 3.3 on page 26) of the answer extraction components. Gaizauskas et al. (2003) showed that the coverage of retrieved documents constantly increases as one descends the IR ranking with a coverage value of 69.2% when considering 200 passages. The performance of their answer extraction component, however, actually decreases from a maximum of 18.6% obtained when using only the first 20 retrieved passages. The document retrieval component is responsible for balancing this trade-off between coverage and answer extraction accuracy.

Of course the total amount of text passed to the answer extraction component can be structured in a number of different ways. If the answer extraction component works better with a small amount of text then it is likely that passing a few full documents will give worse performance than passing a larger number of short relevant passages making up the same volume of text. This is because the second approach is likely to contain more answer instances than the first (assuming that on average the answer to a question appears at most once in any given document – a reasonable but unsubstantiated assumption). A number of papers have reported numerous ways of segmenting full documents to provide passage selection in an attempt to increase the coverage while keeping the volume of text to a minimum. Roberts and Gaizauskas (2004) and Tellex et al. (2003) discuss a number of approaches to passage selection and ranking while Monz (2004) proposes an approach to passage selection based not around fixed passage sizes but rather around the smallest logical text unit to contain the question terms.

5.3 Answer Extraction

Whilst the previous sections should have made it clear that simply processing a question in order to find relevant documents or passages is itself a difficult task it should be remembered that most of the work required to actually answer a previously unseen question takes place within the answer extraction component.

Depending on the approach taken to answer extraction this component will have to perform a number of processing steps to transform the documents passed to it from the document retrieval process into a representation from which the answers can be located and extracted. Note that even though this processing may have been performed in advance and stored along with the document text (see Section 5.2) the differing approaches will be discussed here.

Most answer extraction components rely on the relevant documents being subjected to a number of standard text processing techniques to provide a richer representation than just the words as they appear in the documents. This usually includes tokenization, sentence splitting, POS tagging and named entity recognition. Depending upon the exact approach taken to answer extraction, other techniques will then be applied using the output from these simple techniques.

For example, surface matching text patterns (Soubbotin and Soubbotin, 2001; Ravichandran and Hovy, 2002; Greenwood and Saggion, 2004) usually require no further processing of documents. These approaches simply extract answers from the surface structure of the retrieved documents by relying on a fairly extensive list of surface patterns. For instance, questions for which the answer is a birth date can be answered by extracting answers using patterns such as the following (Ravichandran and Hovy, 2002):

```
<NAME> ( <ANSWER> - )
<NAME> was born on <ANSWER>,
<NAME> was born <ANSWER>
```

```
<NAME> ( <ANSWER> -
```

Whilst assembling extensive lists of such patterns can be time consuming and difficult (see Section 8.2), once assembled they can be exceptionally accurate for such a simplistic approach. One system (Soubbotin and Soubbotin, 2001) which used this approach as its main way of answering questions was in fact the best performing system in the TREC 2001 QA evaluation, correctly answering 69.1% of the questions (official scoring metric was MRR for which this system achieved a strict score of 0.676).

Surface matching text patterns operate surprisingly well using little in the way of NLP techniques other than basic named entity recognition. Semantic type extraction systems, however, require more detailed processing of the documents under consideration. Answer extraction components, such as that described by Greenwood (2004a) operate simply by extracting the most frequently occurring entity of the expected answer type. This requires being able to recognise each entity of the expected answer type in free text, resulting in more complex entity recognition than is required by the surface matching text patterns, usually going well beyond simply named entity detection. While such systems require more detailed processing than the surface matching text patterns they stop short of requiring deep linguistic processing such as full syntactic or semantic parsing of relevant documents.

QA systems such as FALCON (Harabagiu et al., 2000) and QA-LaSIE (Greenwood et al., 2002) make use of deep linguistic processing requiring syntactic and semantic parsing in order to infer connections between questions and possibly relevant documents. To fully appreciate the levels of complexity and processing that these systems undertake we present a simple example of QA-LaSIE finding the answer to *"Who released the Internet worm?"* (Scott and Gaizauskas, 2000). Firstly the question is parsed to produce the QLF representation:

```
qvar(e1), qattr(e1,name), person(e1), release(e2),
lsubj(e2,e1), lobj(e2,e3), worm(e3), det(e3,the),
name(e4,'Internet'), qual(e3,e4)
```

From this QLF representation we can see that the answer that we are seeking (`qvar(e1)`) is a person (`person(e1)`) and that we are looking in particular for the name of the person (`qattr(e1,name)`). We can also see that the person we are looking for should appear as the subject of an instance of the verb release (`lsubj(e2,e1), release(e2)`) which has worm as its object (`lobj(e2,e3), worm(e3)`).

Assuming that a document containing *"Morris testified that he released the Internet worm..."* has been retrieved this parses to produce the following QLF representation:

```
person(e1), name(e1,'Morris'), testify(e2),
lsubj(e2,e1), lobj(e2,e6), proposition(e6),
main_event(e6,e3), release(e3), pronoun(e4,he),
```

```
lsubj(e3,e4), worm(e5), lobj(e3,e5), det(e5,the),
name(e7,'Internet'), qual(e5,e7)
```

Firstly coreference resolution is performed across the text allowing the system to equate the pronoun *he* with the person *Morris*. A discourse model is then built from both the question and the relevant text. From this model the system can determine that the question requests the name of the person who released the Internet worm while the text states that Morris released the Internet worm. This allows the system to correctly determine that *Morris* is the correct exact answer to the question from which a suitable response (depending upon the context and system user) can be generated.

Clearly the approach taken by QA-LaSIE is much more complex and requires more intensive processing than either surface matching text patterns or semantic type extraction and the approaches are vastly different in both the way they tackle the problem and the questions which they are capable of answering. The important point to remember is that while the question analysis and document retrieval components of most QA systems are relatively similar the answer extraction components vary widely and it is upon these that most research into QA has been focused to date.

Chapter 6

Question Analysis

As the first component in a QA system it could easily be argued that question analysis is the most important part. Not only is the question analysis component responsible for determining the expected answer type and for constructing an appropriate query for use by an IR engine but any mistakes made at this point are likely to render useless any further processing of a question. If the expected answer type is incorrectly determined then it is highly unlikely that the system will be able to return a correct answer as most systems constrain possible answers to only those of the expected answer type. In a similar way a poorly formed IR query may result in no answer bearing documents being retrieved and hence no amount of further processing by an answer extraction component will lead to a correct answer being found.

This chapter is divided into two main sections each concerned with one half of the responsibilities of the question analysis component. First a number of approaches, both manual and automatic, to constructing question classifiers are presented and evaluated. The second half of the chapter details a number of ways in which IR queries can be formed and expanded in an effort to improve the documents retrieved by an IR engine, the results of which can be found in Section 7.1.2.

6.1 Determining the Expected Answer Type

In most QA systems the first stage in processing a previously unseen question is to determine the semantic type of the expected answer. Determining the expected answer type for a question implies the existence of a fixed set of answer types which can be assigned to each new question. Determining the set of answer types is in itself a non-trivial problem which can be approached from two different directions 1) answer types which can cover most questions, or 2) a set based upon the answer types a QA system can locate in free text. Both of these approaches will now be discussed in more detail.

In an ideal world QA systems would be capable of answering any question they were asked and for this to happen the set of answer types must be wide enough to cover most

question types. This is the view taken by, among others, Li and Roth (2002) when designing a two-level answer type hierarchy, designed specifically with the typical TREC questions in mind. This hierarchy consists of 6 coarse (top level) categories (abbreviation, entity, description, human, location and numeric value) and 50 fine classes with each coarse class containing a non-overlapping set of fine classes (see Table 6.1, page 61). A similar hierarchy containing 94 nodes is used by the Webclopedia system (Hovy et al., 2000). At first sight these hierarchies seem useful, especially when combined with appropriately tagged examples of each answer type, as systems can be trained to correctly classify previously unseen questions (see Section 6.1.2). Just because a system can classify a question does not, however, immediately mean that the system is capable of answering questions of that type. For example the Webclopedia hierarchy contains a *planet* node signifying that the answer to a question such as *"What is the sixth planet from the Sun?"* should be of type *planet*. If, however, a QA system is unable to determine that a given proper noun is in fact the name of a planet then this answer type serves only to force the system to admit that it is unable to answer the question. This issue was well stated in work by Ittycheriah et al. (2000):

> *"... improvements in the answer type prediction do not correlate directly with improvements in the overall score of the system ... parallel improvements must be made in named entity marking as well as answer selection in order to realize them in the overall system."*

An alternative approach to building a set or hierarchy of answer types is to start by examining the types of answers that the QA system is able to recognise and extract from text (see Chapter 8). Approaching the problem from this direction does of course have its own drawbacks, namely that a different hierarchy has to be constructed for each QA system and in most instances training data is not easily transferable from one system to another.

The answer type hierarchy used throughout the systems detailed in this work (for those systems which require such a hierarchy, specifically the semantic type extraction component detailed in Section 8.1) is shown in Figure 6.1. This hierarchy was constructed using the second approach outlined above starting from the set of entity types that the system could identify and extract. The answer types are arranged in a hierarchy to allow the system to back-off from a specific answer type to a more general type, when required, because some of the types are very narrow and it is not always possible to tag entities with such a fine grained type. For example America's *Arlington National Cemetery* is often referred to simply as *Arlington* which may be tagged as a location (depending upon the context in which it appears) but is unlikely to be tagged with the fine grained type *cemetery*. In this example organizing the types as a hierarchy will allow the system to back-off and return answers of type location if no cemeteries can be found in the text.

It should be clear from the hierarchy given in Figure 6.1 that unlike the hierarchy from Li and Roth (2002) there are many answer types which cover a very restricted set of possible answers. For example the set Planet covers the nine planets of the solar system, Zodiac Signs and Birthstones contain only twelve possible answers and the state mottos,

- Amount
  - Money
  - Percent
  - Measurement
    - Time
      - How Often
      - Age
    - Distance
    - Speed
    - Temperature
    - Area
    - Mass
    - Computer
    - Other
- Currency Unit
- Reward
  - Award
  - Prize
  - Trophy
  - Medal
- Date
  - Month
- Proper Name
- State Motto
- State Bird
- State Flower
- State Tree
- State Nickname
- Organization
- Planet
- God
  - Egyptian
  - Greek
  - Roman
- Person
  - Male
  - Female
- Location
  - City
  - Province
  - Country
  - National Park
  - Lake
  - River
  - Cemetery
  - Continent
  - US State
- Language
- Nationality
- National Anthem
- Religion
- Space Craft
- Job Title
- Quoted Text
- Zodiac Sign
- Birthstone
- Address
- Colour
- Colour List
- Unknown Proper Name
- Chemical Element
  - Symbol
  - Name
- Chemical Compound
  - Symbol
  - Name

Figure 6.1: The answer type hierarchy used for question classification.

flowers, trees, nicknames and birds each contain one answer for each American state. This simply reflects the fact that these types of entity can be recognised and extracted by the QA system and so warrant their own place in the hierarchy. Amalgamating narrow types with their more general class would in fact make answer extraction less reliable due to the increase in noise within the set of entities of the expected type (see Section 3.2.3 for a definition of noise).

6.1.1 Manually Constructed Rules For Automatic Classification

Often the easiest approach to question classification is a set of manually constructed rules. This approach allows a simple low coverage classifier to be rapidly developed without requiring a large amount of hand labelled training data. A number of systems have taken this approach, many creating sets of regular expressions which only questions with the same answer type (Breck et al., 1999; Cooper and Rüger, 2000). While these approaches work well for some questions (for instance questions asking for a date of birth can be reliably recognised using approximately six well constructed regular expressions) they often require the examination of a vast number of questions and tend to rely purely on the text of the question. One possible approach for manually constructing rules for such a classifier would be to define a rule formalism that whilst retaining the relative simplicity of regular expressions would give access to a richer set of features. The rest of this section defines a new formalism and gives both examples of rules and the performance of the resulting classifier.

This classifier contributes to the initial stage of the semantic type answer extraction system of Section 8.1. The performance of this classifier therefore provides an upper-bound on the performance of the answer extraction component as any misclasifications cannot be recovered from.

The rules used by the classifier detailed in this section are formally defined, using Backus-Naur Form (BNF) (Backus et al., 1963), as:

```
rule ::= Rule:[modifier]<name>
         <boolean condition>
         [answer type]

modifier ::= ~|+|-


boolean condition ::= [!]<operator>|[!](<boolean condition>)|
                      <boolean condition>&&<boolean condition>|
                      <boolean condition>||<boolean condition>


operator ::= Rule<name>|string(<x>)|word(<x>)|
             startsWith(<x>)|contains(<answer type>)


answer type ::= NULL|<type>[:<feature>=<value>]*
```

The as yet undefined entities `name`, `x`, `type`, `feature` and `value` are all sequences of one or more ASCII characters. As the BNF definition makes clear each rule can be modified by at most one of the three `modifiers` which have the following effects on the rule being defined:

`~` Marks the rule as a building block only. This means that the rule is not used to classify a question but can be used to build other more complex rules using the Rule*Name* boolean operator described below. As rules modified using this modifier are never used directly to type a question they do not have an associated answer type.

`+` If a question can be classified by multiple rules matching nodes in different branches of the answer type hierarchy then preference is given to a rule modified in this way irrespective of the other selected nodes.

`-` This modifier has the opposite effect from the `+` modifier. Basically this allows the classifier to have very simple rules which match a large number of questions but which are not used if any other rule also matches. For example the classifier currently contains a rule which simply assumes any question containing the word *he* is asking for the name of a man – a very simplistic rule which should be ignored if any other more complex rule can be used to type the same question.

The boolean condition which triggers a rule is constructed from any valid combination (given the BNF definition) of the following boolean operators (which are applied in a case-insensitive fashion to the question text) or their negation (appending an `!` to the beginning of an operator negates the result of the operator):

`Rule`*`Name`*`:` Returns true if the named rule matches the current question, false otherwise. This allows rules to be constructed using the results of other rules which is a useful shorthand for constructing multiple rules which share a common base as it allows the base to be defined only once and then reused.

`word(`*`X`*`):` Returns true if the word *X* or a morphological variant (generated using the Porter stemmer) of *X* appears in the question, false otherwise.

`string(`*`X`*`):` Returns true if the string *X* appears verbatim in the current question, false otherwise. A good example of this would be when classifying a question for which the expected answer is a temperature. Using the operator the rule can check the question for the presence of the string *'boiling point'* or *'freezing point'*, i.e. the rule needs to match a given phrase which is more than a single word in length.

`startsWith(`*`X`*`):` Returns true if the question starts with the string *X*, false otherwise. This is the same as `string(`*`X`*`)` with the extra constraint that the string must be at the start (ignoring white space) of the question.

`contains(`*`Answer Type`*`):` Returns true if the question contains the given answer type, false otherwise. A number of gazetteer and named entity recognisers are run

```
Rule:+Why
word(why)
NULL

Rule:~How
word(how)

Rule:WhereCountry
!RuleWhy && !RuleHow && word(country)
Location:locType=country

Rule:WhoJobHolder
!RuleWhy && !RuleHow && !contains(Person) && contains(Job Title)
Person
```

Figure 6.2: Example rules for determining the expected answer type.

across the question text prior to classification and this operator allows a rule to check for the presence, or absence, of a specific answer type, for instance a Location or Person.

The answer types (and also the argument to the `contains` operator) are actually encoded forms of a document annotation. Before the question classifier is applied the question will have been processed by entity recognisers which will have added representative annotations to the question highlighting all the entities the system could recognise. So, for example, cities will have been annotated as `Location:locType=city` and male names as `Person:gender=male`. If, on the other hand, the answer type for a rule is `NULL`, instead of an actual answer type, then the question is classified as being of an unknown type which causes processing to halt and the system to state that it is not currently able to answer the question. This feature is used, for instance, in a rule which matches against any question containing the word *why* as the system is not designed to provide explanations as answers to questions and so there is no point carrying out any further processing.

To explain the motivations behind this formalism and it's flexibility consider the small set of rules given in Figure 6.2. The first rule simply ensures that the system does not attempt to answer *why* questions. The second rule matches any rule containing the word 'how' and is intended to be used to simplify the building of further rules. The final two rules are built from the previous rules and are designed to recognise and type questions asking for countries or specific job holders. Given these rules the system can attempt to type the following questions which have been subjected to named entity recognition (L is a Location, JT a Job Title, and P a Person annotation):

1. Who is the [US]$_L$ [President]$_{JT}$?
2. Name a past [US]$_L$ [President]$_{JT}$?
3. How long can someone be [US]$_L$ [President]$_{JT}$ for?

4. Which country is [George Bush]$_P$ the [President]$_{JT}$ of?

Whilst question 1 and 2 are asking for different answers the expected answer type for both questions is `Person`. The `WhoJboHolder` rule correctly assigns the type `Person` to questions 1 and 2 as neither contains the words 'why' or 'how' (i.e. neither `RuleWhy` or `RuleHow` match the question) and both contain a known job title but do not contain the name of a person. Question 3 remains untyped given the rules in Figure 6.2 as the only matching rule is `RuleHow` which is a building block only and does not assign an expected answer type. Question 4 is assigned an expected answer type of `Location:locType=country` by the rule `WhereCountry` due to the presence of the word 'country'. `RuleWhoJobHolder` does not match question 4 because of the presence in the question of a person name.

While the example rules and questions have been relatively simple it is hoped they have conveyed the basic approach, and shown how flexible the rules can be in matching questions using very different language but which have the same expected answer type. For example these same rules could also correctly assign the expected answer type `Person` to the question *"Please tell me the name of a previous president of the USA"*. This flexibility in the rules is due to the bag-of-words nature of the rule formalism, i.e. the structure of the question is not important just the words or semantic entities which occur within it.

While a more conventional grammar based approach could have been taken to analyse the question the literature suggests that this would be much more complex without any obvious benefits. For example QA-LaSIE (Humphreys et al., 1999) parses questions using a phrasal grammar hand-built to cover questions. The grammar rules are much more complex than the rule formalism introduced here and attempt to produce a full syntactic and semantic representation of the question. For the purpose of determining the exact answer type this is an overly complex approach and many grammar rules would be required to replicate the power of the example rules given in Figure 6.2. Similarly Paşca (2003) uses a dependency based representation of questions to map the question focus onto a node in WordNet. Sections of WordNet are then associated with expected answer types. Again this approach requires a parser capable of handling questions as well as a manual mapping from WordNet to an answer type hierarchy. The main problem with the approach, however, is that it is limited to handling only those questions with one of the following question stems: what, who, where, how, when, name, which, why, and whom. This specifically excludes processing of questions such as *"Please tell me what the capital of Uruguay is."*. In fact many QA systems (Kupiec, 1993; Breck et al., 1999; Cooper and Rüger, 2000) use an approach based on regular expressions which could be viewed as a forerunner of the approach we have outlined (these systems tend just to use the words in the question and not the presence or absence of semantic entities).

One advantage of approaching the problem of question classification using the rule formalism introduced above is simply that if a question cannot be classified then the system knows that is does not know how to answer it and so can quickly report this to the user. Automatically constructed classifiers (see Section 6.1.2), however, usually classify every instance they are given and so if a question does not actually belong in any of the known

classes it is likely that it will be wrongly classified.

An early version of this question classifier using a hierarchy of 44 question classes (a dummy unknown class was assigned to any unclassified question) was evaluated (single informant evaluation by the author) as part of a run submitted to the question answering track of TREC 2003 (see Appendix D for full details of the run). Of the 413 factoid questions in the test set the classifier assigned an incorrect type to 61 questions of which 27 were assigned no type but should have been assigned a type from the hierarchy. This leaves 34 questions for which answers of an incorrect type were returned. In total 146 questions were of an unknown type and so 267 were typed, 206 correctly (49.88%). If we assume that classifying a question as not having a type is the correct thing to do when it does not belong in any of the known classes then this gives a classification accuracy of 78.69% (325/413) over the 413 questions.

A question classifier containing the 66 classes shown in the answer type hierarchy in Figure 6.1 was evaluated (single informant evaluation by the author) as part of the TREC 2004 question answering track. The question set used in the evaluation contains 230 factoid questions. Of the 230 questions the classifier correctly classified 167 questions, wrongly classified 21 questions and was unable to determine the expected answer type of the remaining 42 questions. Assuming that the unknown class is valid then this gives a classification accuracy over the 230 questions of 90.9% (209/230) into a hierarchy containing 67 classes. It is interesting to note that most of the errors in classification were due to the change in the style of questions from previous TREC evaluations which led to the frequent presence of personal pronouns in the questions (see Appendix E for more details).

6.1.2 Fully Automatically Constructed Classifiers

As mentioned in the previous section building a set of classification rules to perform accurate question classification by hand is both a tedious and time-consuming task. An alternative solution to this problem is to develop an automatic approach to constructing a question classifier using (possibly hand labelled) training data. A number of different automatic approaches to question classification have been reported which make use of one or more machine learning algorithms including nearest neighbour (NN), decision trees (DT) and support vector machines (SVM) to induce a classifier. The major drawback to these approaches is that they often require the extraction of a large number of complex features from the questions, which while feasible during the construction of a classifier may be too intensive to be used for classifying questions in a real-time QA system. For example, Zhang and Lee (2003) describe an SVM based approach to question classification that requires full syntactic parsing of the question being classified. Not all such machine learning approaches suffer from this problem as feature extraction is separate to the learning algorithm being used. For instance, Hacioglu and Ward (2003) describe an SVM based approach to question classification for which they claim that *"... computationally expensive linguistic analysis ... has been avoided"*.

Coarse	Fine
ABBREVIATION	abbreviation, expansion
DESCRIPTION	definition, description, manner, reason
ENTITY	animal, body, color, creation, currency, disease/medical, event, food, symbol, instrument, language, letter, other, plant, product, religion, sport, substance, technique, term, vehicle, word
HUMAN	description, group, individual, title
LOCATION	city, country, mountain, other, state
NUMERIC	code, count, date, distance, money, order, other, percent, period, speed, temperature, size, weight

Table 6.1: Coarse and fine grained question classes.

This section presents a simple approach to question classification which can be quickly applied to unseen questions and utilizes a question hierarchy consisting of 6 coarse grained and 50 fine grained categories as shown in Table 6.1 (Li and Roth, 2002). This hierarchy is used in these experiments, instead of the one previously presented in Figure 6.1, for two reasons. Firstly a large amount of labelled training data is freely available[1] for this hierarchy and secondly a number of other studies (Li and Roth, 2002; Zhang and Lee, 2003) have reported the results of question classifiers developed using this hierarchy and the associated training data, and so the results of these studies are directly comparable to the results obtained here.

The aim of question classification, as has already been stated, is to determine the type of the answers the user is expecting to receive to a given question. Both manually and automatically constructed classifiers work from the same premise that a previously unseen question should be classified based on the known class of similar questions seen during the construction of the classifier. Most of the automatically constructed question classifiers do this by extracting a number of features from a set of questions which have been labelled with the correct answer type (the training data). A machine learning algorithm is then used to induce a classifier from the training data which given a similar set of features, extracted from an unseen question, will assign the appropriate class. These features have included words, POS tags, chunks (non-overlapping phrases), named entities, head chunks (i.e. the first noun chunk in a sentence), semantically related words and syntactic structure. Often these features can be expensive to compute (especially syntactic structures which may require full syntactic and/or semantic parsing of the questions) and so may not be suited for use within real-time QA systems.

A more practical approach to tackling the problem of automatically constructing a classifier would be to use only simple techniques. It is then possible to determine how well such techniques perform compared to the more complex approaches mentioned above. The premise of the approach to question classification adopted here is simple; use an IR engine to index a large number of example questions, which have been labelled with the correct answer type, then given an unseen question select the question or questions most similar to the unseen question assigning the corresponding question type or types (although an actual implementation of this approach will be more complex than this de-

[1] `http://L2R.cs.uiuc.edu/~cogcomp/`

scription suggests). In other words the approach uses an IR engine to perform the job it was designed to do: *given a query (an unseen question) retrieve the documents (labelled questions) which are most like the query*. This approach to question analysis is in-fact an extension of the k-Nearest Neighbour (k-NN) instance-based learning algorithm (Mitchell, 1997). Usually the nearest neighbours of a given instance are defined in terms of the standard Euclidean distance. In this extension of k-NN the neighbours of an instance are defined by the IR engines similarity function.

Before reporting the approach in detail two different precision standards P_1 and $P_{\leq 5}$ are defined. These precision standards can be used to compare the performance with that of other classifiers developed using the same data sets and answer type hierarchy (Li and Roth, 2002).

Suppose for the i-th question k labels are output by the classifier in decreasing order of likelihood then the score $I_{i,j}$ where $0 < j \leq k$:

$$I_{i,j} = \begin{cases} 1 & \text{if } i\text{-th question is correctly answered} \\ & \text{by the label at position } j, \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

Consequently given m test questions $P_{\leq k}$ is defined as:

$$P_{\leq k} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{k} I_{i,j}}{m} \tag{6.2}$$

P_1 (actually $P_{\leq 1}$) corresponds to the usual definition of precision in which each unseen question is assigned a single label, while $P_{\leq 5}$ allows up to five labels to be assigned to each question allowing further stages of a QA system to resolve the remaining ambiguity.

Constructing the classifier involves only the indexing of the 5500 question training set using an IR engine to produce an index in which all question words are subject to stemming (Porter, 1980) and then stored along with their associated coarse and fine grained category labels as provided by Li and Roth (2002). For these experiments we used the Lucene IR engine which is based upon a TF.IDF vector space model, although there is no reason why other term weighting schemes, such as BM25 (Robertson et al., 1994), could not be used instead and should be investigated in any future work. The questions in the training set are not subjected to stopword removal as often the commonly accepted stopwords (i.e. *who, when, where*...) are the most important words in the question for determining the correct answer type.

An examination of the questions in the training set shows that certain words are highly suggestive of a specific question class (or a small subset of question classes), for example *where* nearly always signifies that the expected answer will be some form of location. A list of these words was produced to act as a *'good word'* list – the opposite of a stopword list, in that at retrieval time those words which appear in both this list and an unseen question are considered more influential than other words. The good word list consists solely of the words: *can, city, country, do, first, how, many, there, what, when, where,*

# IR Results	P_1	$P_{\leq 5}$
0	0.0	0.0
1	76.8	76.8
2	76.8	87.4
3	83.8	91.8
4	84.0	94.2
5	84.4	96.0
6	86.8	96.6
7	87.8	97.2
8	88.2	97.6
9	87.8	98.2
10	87.0	98.8
15	86.0	99.6
20	84.0	100.0
25	83.4	100.0
50	82.8	100.0
100	76.6	100.0

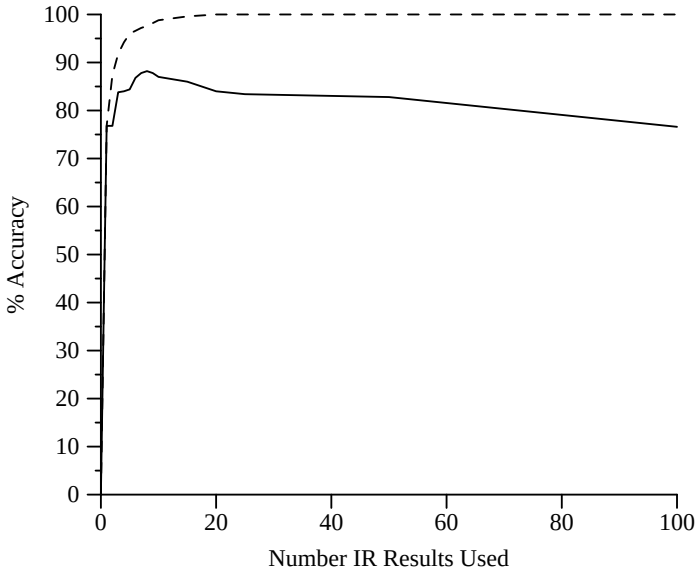

Figure 6.3: P_1 (solid line) and $P_{\leq 5}$ Coarse Grained Classification Accuracy

who, why, you. After brief experimentation these good words were considered to be 6 times more suggestive of the answer type than other words[2].

Determining the class and hence the answer type of an unseen question is treated as a retrieval task. The question is used as an IR query to retrieve the x most similar training questions. For example *"How far is it from Denver to Aspen?"* should be assigned the coarse type NUM and the fine type NUM:dist by the classifier. Using the classifier with x equal to 5 retrieves the following similar training questions and their associated types:

NUM:dist	*"How far is it from Phoenix to Blythe?"*
NUM:dist	*"How far is London UK from California?"*
NUM:dist	*"How far is Yaroslavl from Moscow?"*
NUM:dist	*"How high is the city of Denver?"*
NUM:count	*"How many miles is it to Ohio from North Carolina?"*

The frequency of occurrence of the unique question classes (either coarse or fine) is counted and the question being classified is assigned the most frequent class. For the above example this gives NUM as the coarse class and NUM:dist as the fine grained class, which is a correct classification, on both levels, of the question *"How far is it from Denver to Aspen?"*. The results of using this method to determine the coarse class of the 500 question test set using varied values of x (i.e. varying the number of IR results used to determine the question class) can be seen in Figure 6.3.

It should be clear from the results in Figure 6.3 that the IR based classifier can obtain a $P_{\leq 5}$ accuracy of 100%, for the coarse classification task, which is encouraging given that the best $P_{\leq 5}$ results reported by Li and Roth (2002) is 98.80%. Remember that the

[2] Implementing the *'good word'* list using Lucene simply involves setting the boost of each good word to the required value, 6 in these experiments.

Machine Learning Algorithm	P_1	
	Coarse	Fine
Nearest Neighbour (NN)	79.8%	68.6%
Naïve Bayes (NB)	83.2%	67.8%
Decision Tree (DT)	84.2%	77.0%
Sparse Network of Winnows (SNoW)	86.6%	75.8%
Support Vector Machines (SVM)	87.4%	80.2%
SVM with Tree Kernel	90.0%	$\approx$ 79.2%

Table 6.2: Coarse and fine grained classification performance.

hierarchy into which the questions are being mapped consists of only 6 coarse classes so $P_{\leq 5}$ for the coarse results is equivalent to discarding the least likely class label. It is therefore surprising that the more complex classifier is only able to achieve a $P_{\leq 5}$ of 98.80%. The highest P_1 accuracy of 88.2% obtained in the above experiments is also comparable with the results presented by Zhang and Lee (2003) over the same data using other machine learning algorithms and feature sets, as shown in Table 6.2. It is clear from this table that for a coarse grained classification only the much more complicated SVM with Tree Kernel approach to question classification outperforms this IR based approach. These results also show that, at least for this application, the IR engines similarity function is a better measure of distance between instances than Euclidean distance as the approach outperforms the Nearest Neighbour results detailed by Zhang and Lee (2003).

Considering the fine grained classification results presented in Figure 6.4 it should be clear that the results are not as impressive as those presented in Figure 6.3 for the coarse grained classification but this is understandable given the difference in the number of possible classes to which a question could be assigned (6 classes for the coarse classification and 50 for the fine grained classification task). The highest P_1 accuracy obtained is 77.8% which again is comparable to most of the machine learning approaches detailed in Table 6.2 and outperformed only by the two SVM based approaches[3]. An interesting point to note is that the maximum P_1 accuracy for both the coarse and fine grained classifications is achieved when x is set to 8, i.e. the eight most similar questions are used to determine the classification of unseen questions.

The fine grained classification was approached as a flat classification into 50 separate categories unlike the original hierarchical classification approach (Li and Roth, 2002). Further experiments, however, showed that following the same hierarchical approach of firstly determining the five most likely coarse categories and then classifying using only their fine categories has no effect on the performance of the IR based classifier.

Overall the results of using an IR engine to perform question classification show that it is possible to get relatively good results using simple widely available techniques requiring only the simplest of features to be extracted from the questions both during the construction of the classifier and more importantly when classifying unseen questions.

[3] The original paper (Zhang and Lee, 2003) does not report the fine grained classification results of the SVM with tree kernel but in private correspondence Dell Zhang confirmed that while the results showed a slight improvement to the standard SVM approach the difference was not statistically significant.

# IR Results	P_1	$P_{\leq 5}$
0	0.0	0.0
1	64.2	64.2
2	64.2	76.6
3	72.4	82.6
4	74.6	85.0
5	76.2	86.6
6	77.0	87.4
7	77.2	88.2
8	77.8	88.6
9	77.4	88.8
10	77.6	89.0
15	76.4	90.2
20	74.8	91.4
25	74.0	92.2
50	70.4	90.0
100	63.4	86.2

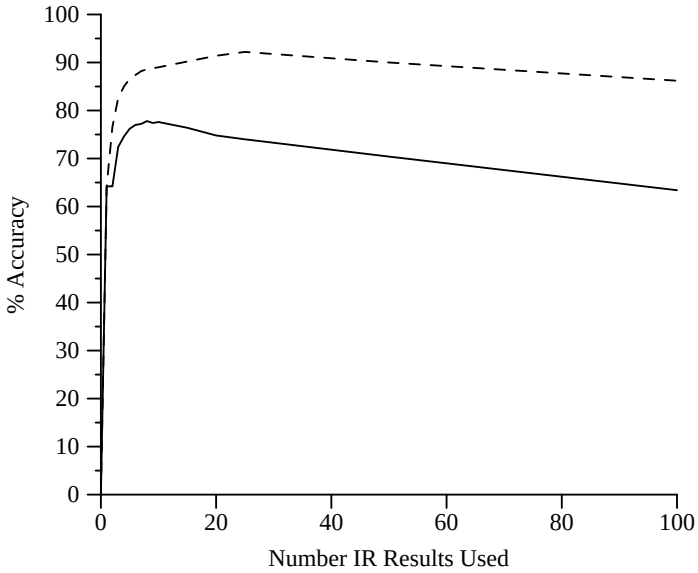

Figure 6.4: P_1 (solid line) and $P_{\leq 5}$ Fine Grained Classification Accuracy

This reliance on only simple features allows questions to be classified quickly, within 5 milliseconds[4], and so lends itself well to use in real-world QA applications.

An interesting extension to these experiments would be to investigate the importance of the term weighting function used by the IR engine. The results presented here were obtained using a basic TF.IDF term weighting. It is possible that utilising a different term weighting scheme, such as BM25 (Robertson et al., 1994), could drastically improve the classifier.

6.1.3 Comparison of Manual and Automatic Approaches to Building Question Classifiers

What is clear from these experiments into question classification, as well as those reported in the literature (Li and Roth, 2002; Zhang and Lee, 2003) is that question classification to determine the expected answer type of a question is a non-trivial problem and should be given due consideration during the development of any sophisticated QA system.

Both the manual and automatic approaches to question classification presented in this section have performed well compared with other reported approaches (Li and Roth, 2002; Zhang and Lee, 2003), with the manually built classifier outperforming the automatically acquired classifier with a precision (P_1) of approximately 90% compared to the 78% achieved by the IR based classifier. While this would suggest that manual question classifiers should be deployed whenever possible it has to be remembered that the amount of time required to construct such a classifier is orders of magnitude more than that required to automatically acquire a classifier.

[4] When run on a PC with a 3GHz Pentium 4 CPU and 1Gb of RAM.

The main advantage of the manual approach over an automatic method is that a manual approach does not require a large amount of manually labelled training data for each expected answer type. Manual classifiers may require many man hours to construct and test to guarantee an appropriate level of question coverage and accuracy but can include rules to cover questions for which only one example has previously been seen. Results presented in this chapter have shown that the manually built classifier has a performance above that of both the IR based classifier introduced in this chapter and the other machine learning approaches reported in the literature. The manually acquired classifier had an accuracy of 90.0% over 67 expected answer types when evaluated using TREC 2004 question set. The best performing automatically acquired classifier (Zhang and Lee, 2003) achieved an accuracy of 80.2% over the 50 fine grained types introduced by Li and Roth (2002).

In the end the choice between manually constructing a classifier or acquiring a classifier from labelled data is likely to come down to the resources available to researchers and the exact task being attempted. For example, real-time QA requires all processing to be carried out quickly and this may exclude some of the more complex features required for some of the best performing machine learning algorithms.

6.2 Query Formulation

The question analysis component of a QA system is usually responsible for formulating a query from a natural language questions to maximise the performance of the IR engine used by the document retrieval component of the QA system.

Most QA systems start constructing an IR query simply by assuming that the question itself is a valid IR query. It should be clear, however, that in some questions many of the words are highly relevant and can be used to locate relevant documents while in others few words are shared between the questions and possible realisations of the answers. For example *"How tall is Mt. Everest?"* contains only one relevant phrase *"Mt. Everest"*. The problem with questions such as this are that documents containing the answer are unlikely to be worded in a way in which the other words in the question (in this case *tall*) can be of any help, for example *...at 29,035 feet the peak of Mt. Everest is the highest point on Earth*.

Two possible approaches can be taken to expanding queries to help retrieve relevant documents based upon either the expected answer type or expansion of the words in the question. Both of these approaches are examined in the following sections.

6.2.1 Query Expansion via Expected Answer Type

One possible solution to the problem of mismatch between natural language questions and documents containing their answers would be to produce a semantically based index of the document collection. For each document such an index would list whether or not

Answer Type	Expansion Terms
Measurement:type=computer	exabit, petabit, terabit, gigabit, megabit, kilobit, bit, exabyte, petabyte, terabyte, gigabyte, megabyte, kilobyte, byte, octet, KB, MB, GB, TB
Measurement:type=distance	yard, feet, inch, metre, centimetre, millimetre, kilometre, mile, cubit, furlong, yd, ft, in, m, cm, mm ,km
Measurement:type=mass	ounce, pound, gram, kilogram, kg, oz
Measurement:type=temperature	degrees, Celsius, centigrade, Fahrenheit, Kelvin, K, °C
Measurement:type=time	minute, second, hour, year, day, week, month, min, s

Table 6.3: Query expansion terms for a number of answer types.

the document contained instances of the semantic entity types about which the QA system is aware (see Section 6.1), for instance measurements, locations, the names of people and organizations, etc. Having created such an index a system could, for a given question, locate all those documents which contain at least one instance of the answer type and are deemed relevant by a standard IR system. The problem with this approach is that a separate index of answer types would have to be constructed for every collection within which we would like to find answers – something which is not always feasible although in the future the semantic web (Berners-Lee, 1998) may make this a possibility for systems which use the web as a document collection.

An alternative approach would be to expand the IR query to include terms which are likely to cause more documents containing entities of the required type to be returned. This approach is well suited to answering certain question types, including those asking for a measurement, which can be recognised using a fixed set of keywords. For example distance measurements will often be associated with words such as *feet*, *inches*, *metre*[5], *centimetre*.... In fact this is one way in which systems can extract measurements from free text to use as possible answers to questions (see Chapter 8). Table 6.3 gives relevant words for a selection of measurement types.

Both of these approaches, regardless of their performance, cannot currently be applied to document retrieval over the world wide web. Firstly it is infeasible, certainly in the context of this research, to index the entire web recording which semantic entity types a given page contains. The second approach also cannot be applied to the web as most search engines limit the number of search terms per query[6]. We can however easily apply the second approach[7] to closed collections, such as AQUAINT, over which we have complete control. The results of applying such an approach to query formulation on the documents retrieved by the IR component of a full QA system, for questions whose expected answer type is a measurement, are given in Section 7.1.2.

[5] In the experiments American as well as British spelling is used as the majority of the documents in the AQUAINT collection are written in American English.

[6] The Google Web API, used extensively throughout the research documented in this thesis, limits queries to a maximum of 10 search terms.

[7] It is of course also feasible to generate an index of people, measurements etc. in a closed collection but this requires a more complex index structure and would also require re-processing the document collection if any new answer types are added to the system or if the approach to entity detection changes.

6.2.2 Query Expansion via Question Words

An alternative to query expansion via expected answer type is to expand queries based upon the words in the question. A number of approaches to expanding queries in this fashion, including synonym expansion, have been reported (Harabagiu et al., 2000; Hovy et al., 2000). The main problem associated with this form of expansion is that many words can have more than one meaning and so need to be disambiguated before they can be expanded. Also expansion of all terms in the query can lead to large complex queries which then become difficult to refine. The approach to query expansion presented in this section is a selective approach that aims at improving the IR queries for a specific set of questions.

Upon examining a number of questions and the associated documents which are known to contain correct answers, it is clear that while a large number of questions ask for information about a specific location mentioned in the question, i.e. *"What is the state bird of Alaska?"* the answers frequently occur with the adjective form of the location, i.e. *"... willow ptarmigans (the quail-like Alaskan state bird)"* and without the noun form appearing within the relevant passages. Most other question words, however, appear unaltered in the relevant passages, e.g. *state* and *bird* appear in both the question and supporting documents, in fact it is the presence of these words which make the documents relevant.

This obvious mismatch between the question and answer texts is likely to mean that relevant documents are either not retrieved or are lowly ranked by most IR engines. Even those IR systems which employ stemming in an attempt to retrieve documents containing morphological variants of question words are unlikely to fare any better as most adjective forms of a location do not produce the same token when stemmed as is produced by stemming the noun form of the location – a notable exception being *Philippines* and *Philippine* which are both stemmed to `philippin` using the well known Porter stemmer (Porter, 1980). It should be noted that the Porter stemmer was developed over the Cranfield 200 collection (Cleverdon and Keen, 1966) of technical aeronautical documents and as such the ability to conflate location nouns and adjectives was unlikely to have been considered.

WordNet (Miller, 1995) contains pertainym relations which link adjectives with their associated nouns (i.e. *Alaskan* to *Alaska*) and mining this information allows the inverse mapping from nouns to adjectives to be determined (note that while the adjective to noun mapping in WordNet is a one-to-one relationship the noun to adjective mapping is a one-to-many relationship). Together these mappings allows us to experiment with expanding both location nouns and adjectives (i.e. *Alaska* can be expanded to *Alaskan* and vice versa) to form richer IR queries from the original natural language questions. The advantage of using only a specific WordNet relationship over unambiguous questions words is that the system does not need to perform word sense disambiguation prior to expansion, the query does not become excessively complex and the reason for expansion is grounded in the reality of written English (or at least that which occurs in the AQUAINT collection).

There are a number of ways in which pertainym relationships could be used to expand IR queries with the aim of improving the quality of the retrieved documents. The simplest

approach is to just include the two alternative forms of the location within the IR query by replacing the original term with a nested `or` query, for example.

"*What is the capital of Syria?*" becomes

```
capit (syria syrian)
```

Unfortunately this may result in unwanted side-effects when the retrieved documents are ranked based upon their similarity to the query. While `or` has a single interpretation when used to retrieve documents from a collection, documents which contain either or both terms are retrieved, it can be used be used in a number of ways when the retrieved documents are ranked.

When ranking documents retrieved using a query of the form `A or B` ranking functions can consider matches against `A` and `B` as separate matches in which case those documents which contain both will rank higher than those which just contain one (or even two instances of one). The alternative is for the ranking function to treat the combination as a whole and hence those documents which contain a single instance of `A` or `B` will be treated the same as a document which contains both, i.e. the combination matched so we do not care that both `A` and `B` matched only that `A` or `B` matched. Most IR engines adopt the first approach to ranking in that a document which matches more of the query terms will rank highest regardless of the fact that the `or` operator specifics that only one has to match.

For instance the Lucene IR engine[8] defines the similarity between a query q and a document d as follows. Let $f_{t,d}$ be the frequency of term t in document d, n_t be the number of documents in which t occurs in a collection of N documents and l_x be the number of terms in the text fragment x then the similarity between q and d is given as[9]:

$$similarity(q,d) = \frac{l_{q \cap d}}{l_q} \sum_{t \in q} l_d \sqrt{f_{t,d}} log\left(\frac{N}{n_t}\right) + 1 \tag{6.3}$$

Now there are two important things to notice about this ranking function. Firstly the term independent weight, $\frac{l_{q \cap d}}{l_q}$, is simply the percentage of the query terms which appear in the document. For a query `A or B` ranking a document which contains `A` and `B` will compute a weight of 1, while a document which contains only `A` or `B` the weight will be 0.5. Secondly each term is scored independently based upon the term's collection frequency, n_t. As it is highly likely that two synonymous terms `A` and `B` will have different frequencies of occurrence across the document collection, the one which appears in a given document may again affect the relative ranking of the retrieved documents.

A better approach to incorporating the idea of synonymy within an IR query is to treat both `A` and `B` as alternative versions of each other. For this purpose we introduce the IR operator `alt`. The `alt` operator results in the same documents being retrieved as when using the `or` operator although the documents are treated differently by the ranking function. The ranking function now treats `A` and `B` as instances of the same term hence the

[8] `http://jakarta.apache.org/lucene`

[9] This is a simplified version of the full similarity equation as it does not perform any normalization to allow results from different queries to be compared. This simplification does not, however, affect the ranking of documents for a given query.

term independent weight is now 1 even if the document only contains A or B. The `alt` operator also alters the per-term section of the similarity function by using n_A for n_t even if the document only contains instances of B. Together these two changes to the ranking function cause the IR engine to treat A and B identically.

This new operator results in IR queries which give equal weight to the noun and adjective forms of the same location, such as:

"What is the capital of Syria?" becomes

```
capit alt(syria, syrian)
```

Logic would suggest that if either operator was to result in an increase in the quality of the retrieved documents (i.e. an increase in the answer coverage of the retrieved documents) then the `alt` operator should give the better performance of the two options. The results of experiments to apply both approaches to the task of expanding location nouns with their adjectival forms and vice versa can be found in Section 7.1.2.

This section has introduced two ideas; selective expansion of question words and the `alt` operator for ranking retrieved documents. While we have only used them both to deal with questions containing a location noun there is no reason why either other sets of words could not be selectively expanded or why the `alt` operator could not be used in other situations. For example, Bilotti et al. (2004) suggest expanding question words with their morphological variants rather than using stemming when building and retrieving documents from an indexed collection. It would be interesting to see if the `alt` operator could benefit this approach by grouping the morphological variants of a single word. The selective expansion of query terms could also be investigated further although it is unclear what sets of words would benefit from expansion without first performing word sense disambiguation to ensure the correct senses were expanded. Previous work by Voorhees (1993) showed that WordNet could be used to improve IR performance when word sense disambiguation was correctly performed but on average errors in sense resolution result in a decrease in IR performance. The advantage of our approach is that we do not need to perform word sense disambiguation and so should not suffer the negative affects described by Voorhees (1993).

6.3 Summary

Question analysis is an important component of many modern QA systems performing two main functions: expected answer type determination and query formulation. This chapter has presented contributions to both aspects of question analysis.

While others (Ittycheriah et al., 2000) have noted that increases in answer type prediction do not guarantee improvements in answer extraction, this chapter has introduced the notion that the problem should in fact be approached in reverse – from the semantic entities the system can extract. The answer type hierarchy in Figure 6.1 was therefore constructed to contain only semantic types which our answer extraction component (see Chapter 8) can recognise in free text. Implicit in this approach is the notion that if the expected an-

swer type of a given question is not present in the hierarchy then the QA system knows before any further processing that it will not be able to answer the question. This allows the QA system to respond quickly to the user when it cannot answer a question.

Given an answer type hierarchy, the question analysis component must be able to assign answer types to unseen questions. Two new classifiers have been introduced and evaluated in this chapter. Firstly we introduced a rule formalism for the manual construction of a classifier. The flexibility of the formalism allows simple rules to correctly type a large number of questions. In an independent evaluation over the TREC 2004 question set a classifier constructed using this formalism had a classification accuracy of 90.9% (using the hierarchy in Figure 6.1). The chapter reported the results of using an IR engine to automatically construct an answer type classifier. Using the answer type hierarchy (50 fine grained classes) introduced by Li and Roth (2002) this classifier achieves an accuracy of 77.8% outperforming a number of standard machine learning algorithms applied to the same problem (Zhang and Lee, 2003).

The final section of this chapter detailed two novel approaches to query formulation: selective expansion of query words and the `alt` ranking operator. While we have only used selective expansion for a small set of questions containing a location, it is hoped that the idea can be extended to cover sets of words which could also be unambiguously expanded. Query expansion is of course only useful if it provides an improvement in the answer coverage of the documents retrieved by an IR engine. For this reason we propose a new `alt` operator to alter the normal ranking algorithms of TF.IDF based vector space IR models which tend to prefer documents containing multiple forms of the same word. Evaluations of both these ideas are presented along with other work on document retrieval in the next chapter.

Chapter 7

Document Retrieval

Finding exact answers to unseen questions requires the detailed processing of free text. The document collections over which QA systems operate tend to be so large that carrying out such processing of all documents in the collection is not feasible. This has led to most researchers using off-the-shelf retrieval systems in order to select a small subset of documents which can then be subjected to more detailed processing by the answer extraction component of the QA system. It is of course possible to employ novel indexing or retrieval methods to this stage in a QA system to increase the quality of the retrieved documents, especially if dealing with reasonably sized closed collections.

This chapter details the evaluation of a number of novel approaches to retrieving relevant documents or passages from both closed collections and large ever changing collections such as the web. This includes evaluating the performance of the different methods of query formulation detailed in Section 6.2. The performance of these approaches to retrieval are evaluated using the metrics defined in Chapter 3.

7.1 Document Retrieval Over Closed Collections

Working with closed document collections, such as the AQUAINT collection, makes it possible to experiment with many different approaches to indexing and query formulation. All the work presented in this thesis relies upon the Lucene IR engine[1]. Lucene is an open source boolean search engine with support for ranked retrieval results using a TF.IDF based vector space model. One of the main advantages of using Lucene, over many other IR engines, is that it is relatively easy to extend to meet the demands of a given research project (as an open source project the full source code to Lucene is available making modification and extension relatively straight forward) allowing experiments with different retrieval models or ranking algorithms to use the same document index.

While different retrieval approaches can be used on top of a single document index, the creation of the index itself should be given due consideration. The original source doc-

[1] `http://jakarta.apache.org/lucene/`

uments have to be processed, at the word level, to generate a representative index of the collection allowing the option of applying any number of algorithms to the text. For instance stopwords are usually not included in the index as they are of little or no help at retrieval time while greatly enlarging the size of the index. The other main decision to be made when indexing documents is whether or not to normalise variants of a word or term so that searching for one will match against all the known variants. Many systems make use of stemmers (such as the Porter (1980) stemmer) to merge morphological variants of a word under a single index entry. While this seems to be the normal approach to document indexing it should be noted that Bilotti et al. (2004) claim that performing stemming reduces the ability of IR engines to retrieve relevant documents. They suggest that IR queries should instead be expanded to include all the morphological variants of the question words. In the experiments that follow all the documents in the AQUAINT collection were split into separate paragraphs (based upon the original markup which leads to passages of varying lengths) which were then subjected to both stopword removal and stemming before being indexed by Lucene.

7.1.1 How Much Text Should be Retrieved and Processed?

One of the main considerations when doing document retrieval for QA is the amount of text to retrieve and process for each question. Ideally a system would retrieve a single text unit that was just large enough to contain a single instance of the exact answer for every question (and possibly a slightly longer piece of text which also includes enough context to justify it). Whilst the ideal is not attainable, the document retrieval stage can act as a filter between the document collection and answer extraction components by retrieving a relatively small selection of text excerpts (not full documents) from the collection which can then be passed to an answer extraction component for further, more detailed, processing. This is the rational behind the full documents being split into passages (using the existing paragraph markup) during the creation of the index. During the rest of this discussion passages and not full documents are being retrieved from the collection for evaluation and further processing.

In an attempt to determine the amount of text to retrieve for each question a number of evaluations were carried out to investigate the performance of an answer extraction component (see section 8.1) when presented with a varying number of text passages. As discussed in Chapter 4, the 444 questions from TREC 2002, which are known to have at least one correct answer in the AQUAINT collection, are used in these experiments.

Figure 7.1shows the results of these experiments from which it is possible to determine a number of interesting and useful facts which will dictate the way in which a document retrieval component can be configured for use within a QA system.

The results in Figure 7.1 show that the accuracy of the answer extraction component reaches an $a@1$ peak when using at most 20 documents from the collection to answer each question even though the coverage, $c@n$ continues to increase (see Section 3.1.1 for a definition of $a@n$). Considering more than 20 documents actually decreases the perfor-

	Number of Passages							
	1	**5**	**10**	**20**	**30**	**50**	**100**	**200**
$a@1$	8.11%	16.22% ▲	19.14% ▲	20.95%	20.27%	19.14%	18.47%	16.67% ▽
$a@2$	8.11%	19.59% ▲	24.55% ▲	26.13%	27.92% △	28.15%	26.80%	26.58%
$a@3$	8.33%	21.40% ▲	26.80% ▲	29.73% ▲	31.08%	32.66%	29.95% ▼	28.83%
$a@4$	8.33%	22.30% ▲	27.93% ▲	32.21% ▲	32.43%	34.68% △	33.11%	31.08% ▽
$a@5$	8.33%	22.52% ▲	28.38% ▲	34.46% ▲	34.91%	36.26%	33.56% ▼	31.53% ▽
$c@n$	11.04%	29.95%	39.41%	51.35%	56.98%	61.71%	67.34%	71.85%

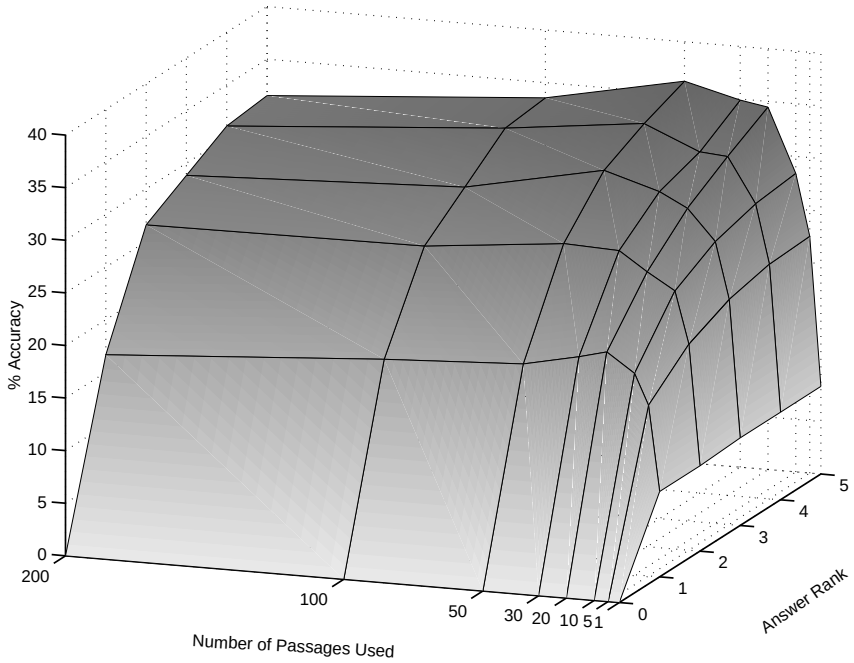

Figure 7.1: Accuracy against number of passages retrieved and number of answers considered (statistically significant differences in performance are with respect to previous column).

mance of the answer extraction component. It should be noted that the $a@5$ performance continues to increase and reaches a peak when using the top 50 documents. However, an ideal QA system would return a single exact answer and so the $a@1$ performance is of more importance. The drop off in performance as more documents are considered by the answer extraction component is understandable given that the answers to the questions are judged using the pooled responses from the TREC 2002 evaluation for which on average each question can be answered by text found in only 1.95 documents from the AQUAINT collection. As was mentioned in Chapter 3 these judgments are not exhaustive and so there are in fact significantly more relevant documents in the collection than the judgments suggest. Even an exhaustive search of AQUAINT to locate answer bearing documents (Bilotti et al., 2004) suggests that the average number of relevant documents per question is actually 15.84 from a collection containing approximately 1,033,000 documents. With so few relevant documents, not all of which will be successfully retrieved by an IR system, returning a large number of documents for the answer extraction component to process will simply result in adding more noise to the text making the task of extracting the correct answer that much harder.

As already discussed the results in Figure 7.1 show that the best $a@1$ performance of

	$c@n$	$a@1$
n = **1**	11.04%	8.11%
n = **5**	29.95%	16.22%
n = **10**	39.41%	19.14%
n = **20**	51.35%	20.95%
n = **30**	56.98%	20.27%
n = **50**	61.71%	19.14%
n = **100**	67.34%	18.47%
n = **200**	71.85%	16.67%

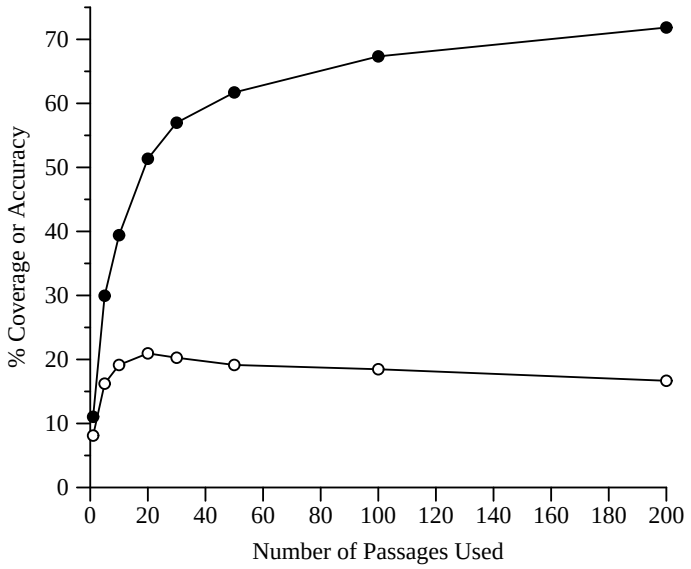

Figure 7.2: Comparison of coverage, •, against accuracy of one exact answer per question, ○.

the answer extraction component occurs when just the top 20 passages are considered by the answer extraction component. Whilst using only a small number of documents is desirable it should be noted that using a small number of documents tends to result in low coverage and hence a lower upper bound on the performance of the QA system. The coverage and accuracy for $a@1$ are shown in Figure 7.2 from which it can be seen that at rank 20, when $a@1$ is maximised, the coverage is only 51.35%, that is the upper bound on a perfect answer extraction component would be 51.35%.

While it is clear that if each question is only correctly answered by a very small percentage of the documents in a collection then retrieving more documents is unlikely to improve the performance of the answer extraction component it would be useful to measure the increase in noise as more documents are processed, where we define noise to be the proportion of the entities of the same type as the expected answer which do not correctly answer the question (see Section 3.2.3 for a formal definition of noise). Figure 7.3 shows how the noise level of the set of retrieved documents increases as the number of documents examined also increases. As you can see when an answer extraction component examines over 20 documents it has to deal with a noise level of approximately 85% – that is, of all the entities in the documents which are of the expected answer type only 15% will be correct answers to the question (where the correctness of an answer is determined using the accepted regular expression patterns as detailed in Section 3.1.1). This reinforces the need to produce a document retrieval approach that while achieving high coverage does so at low ranks.

While these results are certainly interesting it is unclear how transferable the conclusions are to either other IR systems or other answer extraction components. It should be noted, however, that almost identical coverage against accuracy results were reported for the QA-LaSIE answer extraction component (Gaizauskas et al., 2003) and using the Okapi IR engine[2](Robertson and Walker, 1999) results of which can be seen in Figure 7.4. This

[2] `http://www.soi.city.ac.uk/~andym/OKAPI-PACK`

	n@n	a@1
n = **1**	72.95%	8.11%
n = **5**	77.34%	16.22%
n = **10**	79.63%	19.14%
n = **20**	84.43%	20.95%
n = **30**	86.13%	20.27%
n = **50**	88.44%	19.14%
n = **100**	90.84%	18.47%
n = **200**	92.31%	16.67%

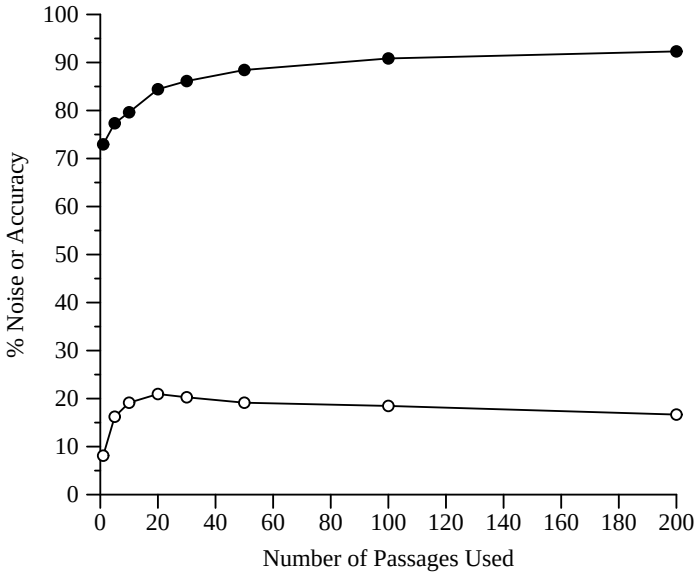

Figure 7.3: Comparison of document noise, ●, against accuracy of one exact answer per question, ○.

combination also gave the best a@1 performance when Okapi was used to retrieve the top 20 passages of a similar size to those used in the experiments reported above. In fact the a@1 performance was 18.59% (compared to the result of 20.95% reported earlier) for which the c@20 was only 54.0% (compared to the result of 51.35% using Lucene). Similar findings using web search results were reported by Dumais et al. (2002) in which the end-to-end performance (measured using MRR, see Section 3.1) of their QA system peaked when using only the first 200 snippets after which the addition of more snippets reduced the systems performance.

Numerous studies have concentrated on the ability of IR engines to retrieve relevant documents in answer to natural language questions and all have shown that no system retrieves documents for every question even if a vast number of documents are considered (Tellex et al., 2003; Roberts and Gaizauskas, 2004; Saggion et al., 2004). For example Hovy et al. (2000) report that even when considering the top 1000 documents per question their IR setup failed to retrieve a single answer bearing document for 8% of the questions. What is required therefore are approaches to both document and passage retrieval that not only increase the coverage of retrieved documents but which also result in needing to retrieve only a few documents per question which can then be processed by the answer extraction components. Systems need to focus on a small number of documents as the results reported here have shown that processing more documents (while increasing the coverage) tends simply to swamp answer extraction components decreasing the end-to-end performance of the QA systems.

The rest of this section details a number of approaches to information retrieval which aim to achieve both of these things, while not overly increasing the time or resources required to retrieve the documents, so as to keep the system useable in real-time.

	$c@n$	$a@1$
n = **1**	18.6%	11.55%
n = **5**	37.0%	16.06%
n = **10**	46.0%	17.75%
n = **20**	54.0%	18.59%
n = **30**	57.4%	16.62%
n = **50**	61.4%	17.18%
n = **100**	66.8%	16.06%
n = **200**	69.2%	12.68%

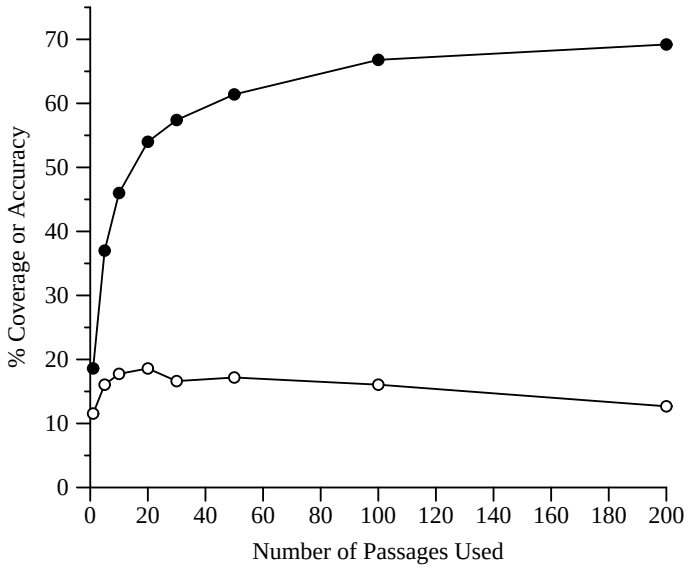

Figure 7.4: Comparison of Okapi coverage, •, against accuracy of QA-LaSIE returning one exact answer per question, ○.

7.1.2 Benefits of Query Expansion

This section reports the results of the query formulation ideas detailed in Section 6.2 and their impact upon the performance of the IR component itself and the effects, if any, on the performance of the end-to-end system.

Expansion via Expected Answer Type: As explained in Section 6.2.1 it is possible to form IR queries using the natural language question and keywords known to commonly occur with instances of the expected answer type. Due to a limited number of questions for which answers are known to be contained within the AQUAINT collection we conducted experiments using this approach for only three answer types; distance, time and temperature measurements. These evaluations use appropriate questions from the TREC 2002 and 2003 evaluations; 33 distance questions, 4 time questions, and 7 temperature questions. Due to the small number of questions it will be difficult to show that results are statistically significant but we should get some notion of the performance of the approach.

For each of the three answer types, distance, time and temperature, both the coverage of the IR results and the accuracy of the end-to-end system performance (using the semantic type extraction system detailed in 8.1) were measured for three different forms of IR query construction:

- **Standard:** An IR query was constructed using only the non-stopwords from the question.
- **`or` Expansion:** An IR query was constructed using the non-stopwords in the question augmented with the appropriate expansion terms (see Table 6.3) combined using the `or` operator, which means that each document retrieved must contain at least one of the expansion terms.

	Distance			Time			Temperature		
	Std	**or**	**alt**	**Std**	**or**	**alt**	**Std**	**or**	**alt**
c@1	3.0%	12.1% △	15.2% △	25.0%	25.0%	50.0%	14.3%	14.3%	14.3%
c@5	18.2%	27.3%	42.4% △	25.0%	75.0%	75.0%	28.6%	28.6%	14.3%
c@10	27.3%	51.5% ▲	57.6% ▲	75.0%	75.0%	100.0%	28.6%	28.6%	14.3%
c@20	45.5%	63.6% ▲	72.7% ▲	100.0%	100.0%	100.0%	42.9%	28.6%	14.3%
a@1	12.1%	12.1%	18.2%	25.0%	25.0%	25.0%	14.3%	14.3%	14.3%
a@2	21.2%	21.2%	24.2%	50.0%	25.0%	50.0%	14.3%	14.3%	14.3%
a@3	27.3%	33.3%	33.3%	50.0%	50.0%	50.0%	14.3%	14.3%	14.3%
a@4	30.3%	36.4%	36.4%	50.0%	50.0%	50.0%	14.3%	14.3%	14.3%
a@5	33.3%	36.4%	36.4%	50.0%	50.0%	50.0%	14.3%	14.3%	14.3%
n@20	85.5%	87.0%	89.6%	64.6%	85.4%	80.3%	85.7%	85.7%	85.7%

Table 7.1: Results of expanding distance, time and temperature questions (statistical significance is with respect to the standard query).

- **`alt` Expansion:** This IR query retrieves the same documents as the `or` expansion query but the documents are ranked in a more sensible fashion which does not take into account the number of expansion terms common to both the query and document. See Section 6.2.2 for details of the .

The results of these experiments into query expansion are presented in Table 7.1. These results show that neither method of query expansion results in a significant change in the end-to-end performance of the QA system for any of the three answer types. The answer accuracy of questions requesting a distance measure show a slight improvement, temperature questions show no change and time questions show a slight drop in performance (a@2 is reduced by 50%. using the `or` expansion method). Due to the small number of questions used in these experiments it is difficult to draw any strong conclusions concerning the use of the `or` and `alt` expansion operators. While there appears to be a considerable increase in coverage using `alt` for the distance questions, the reverse is true for the temperature questions.

Table 7.1 shows that while expanding the queries often produces a dramatic increase in coverage of the retrieved documents at a given rank, the search space within which the answer extraction component has to operate is also increased. The noise level of the retrieved documents, the percentage of the entities of the expected type which are not correct answers to the question (see Section 3.2.3), is increased when expanding the distance and time queries making it harder for an answer extraction component to determine which of the entities of the correct type correctly answers a given question.

From these evaluations of query expansion it is unclear if there is any benefit to be gained from performing query expansion based upon the expected answer type. In some cases the performance of the IR component, as measured by coverage, increases while in others there is little or no positive improvement. Even for questions requesting a distance for which the increase in coverage is statistically significant there is no corresponding significant increase in the end-to-end performance of the QA system. This suggests that not only is this form of query expansion not suitable for question answering but also that coverage should only be used as a guide to measure IR performance; an increase in coverage (for a

	Expansion Type		
	none	or	alt
c@1	15.8	14.0	14.0
c@5	31.6	31.6	36.8
c@10	38.6	36.8	43.9
c@20	52.6	43.9	57.9
c@30	54.4	50.9	64.9 ▲
c@50	63.2	57.9	66.7
c@100	71.9	68.4	75.4
c@200	78.9	75.4	84.2 △

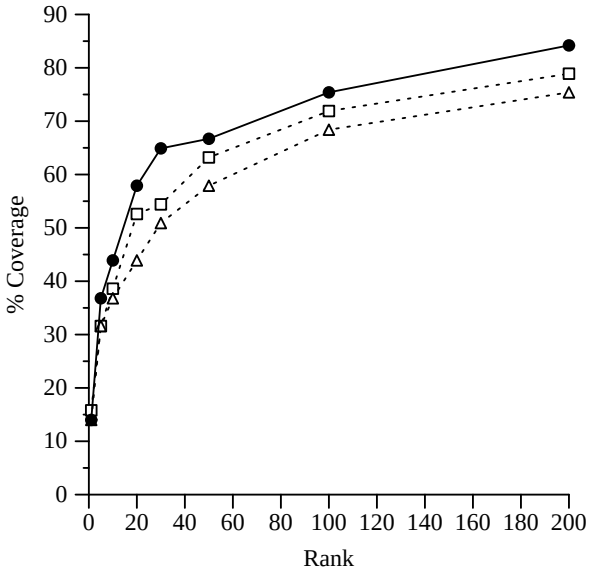

Figure 7.5: Comparison of standard queries, □, with `alt`, •, and `or`, △, expansion of location nouns.

given volume of text) does not imply an increase in end-to-end system performance.

Monz (2003) also reports work on a similar approach to query expansion for measurement questions. The research differs from that presented here in that all measurement questions were evaluated as a whole rather than evaluating different types (i.e. distance, time...) separately. These evaluations were also only concerned with the performance of the IR component and not how the retrieved documents would affect the end-to-end performance of a full QA system. Monz concluded that while this form of query expansion gives improvements in coverage the difference is rarely significant and so is unlikely to contribute greatly to the performance of an answer extraction component making use of the retrieved documents.

Expansion via Question Words Two separate evaluations were carried out to determine the performance benefits of expanding queries using location pertainyms mined from WordNet using the method described in Section 6.2.2. In the first we expanded questions which contain location nouns to also include the adjective form. In the second evaluation we expanded questions containing the adjectival form of a location to include the nominal form of the location. The question sets for these evaluations consist of 57 questions containing location nouns and 31 questions containing adjectival locations taken from the TREC 2002 and 2003 questions sets.

The result of expanding queries to include adjectival forms of locations contained in the original questions can be seen in Figure 7.5[3].

[3] Previously reported results (Greenwood, 2004b) for this approach contained an error in the evaluation of expansion using the `or` operator and differ dramatically from the correct results presented here. The errors in the evaluation of the `or` operator caused many relevant documents to be excluded from the previous evaluation.

	Expansion Type		
	none	or	alt
c@1	22.6	22.6	19.4
c@5	41.9	45.2	38.7
c@10	54.8	48.4	51.6
c@20	64.5	64.5	61.3
c@30	71.0	64.5	67.7
c@50	80.6	71.0 ▽	74.2
c@100	83.9	77.4	80.6
c@200	87.1	77.4 ▽	83.9

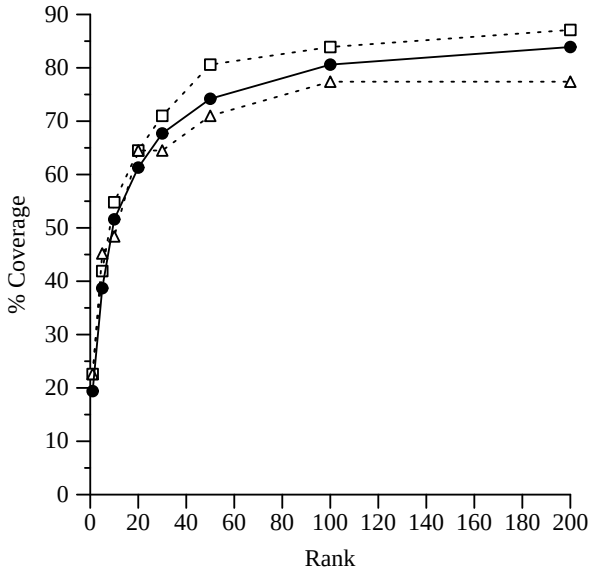

Figure 7.6: Comparison of standard queries, □, with `alt`, •, and `or`, △, expansion of location adjectives.

These results show that the coverage of the retrieved documents increases when the question is expanded to include the adjectival forms of the locations using the `alt` operator. The difference is only significant, however, when 30 or more documents are considered, although this could be due at least in part to the relatively small size of the question set (only 57 questions). It is obvious from these results that using the standard `or` operator to expand the queries has a severe detrimental effect on the results. As has already been discussed this is due to the fact that answer bearing passages tend to contain only one form of the location and using a ranking system that prefers documents which contain both forms pushes answer bearing passages much further down the ranking.

The results of the second evaluation, to determine whether or not expanding adjective locations in questions to include the noun form of the location has an appreciable benefit on the coverage of the retrieved documents, can be seen in Figure 7.6. This experiment shows that over the 37 questions containing an adjective form of a location the coverage of the retrieved documents is actually reduced when the location noun is included in the query, although the drop in performance is not significant at any rank examined. A larger test set is required to see if the observed drop in performance is true in general or simply an artifact of the current question set. These results also confirm the results from the first experiment that using the `or` operator to expand natural language questions into IR queries has a detrimental affect on the ability of the IR engine to rank highly the known answer bearing documents.

Due to the apparent drop in performance observed in the second experiment when including the location in the IR query a third experiment was undertaken. This third experiment used the first question set (which contains location names) and replaced all the locations with their adjective form rather than expanding the IR queries to include both forms. The motivation behind this experiment is that while including the adjective form in the first experiment produced an increase in coverage adding the nominal in the second experiment reduced the coverage suggesting that the adjectival form may be solely responsible

	Expansion Type	
	none	replacement
c@1	15.8%	10.5%
c@5	31.6%	29.8%
c@10	38.6%	33.3%
c@20	52.6%	49.1%
c@30	54.4%	50.9%
c@50	63.2%	57.9%
c@100	71.9%	63.2%
c@200	78.9%	64.9% ▽

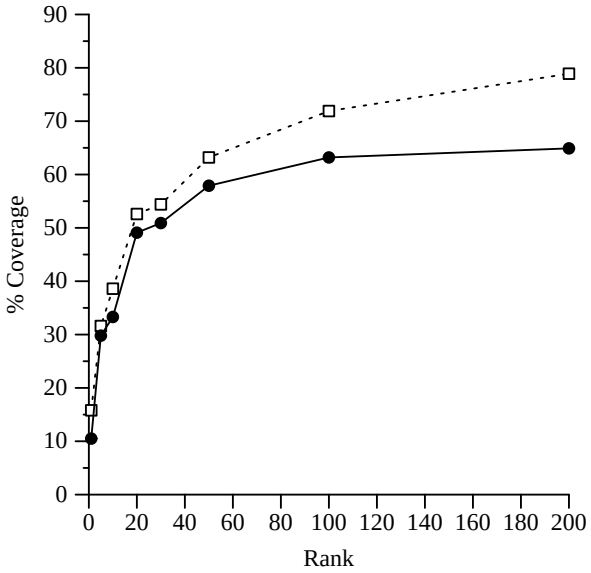

Figure 7.7: Comparison of standard queries, □, and those in which nouns are replaced by adjectives, ●.

for good IR performance. Queries generated in this experiment include:

"What is the area of Venezuela?" becomes

```
area venezuelan
```

"What continent is Scotland in?" becomes

```
contin scottish
```

The results of this experiment can be seen in Figure 7.7. These results suggest that the best way to locate relevant documents for questions containing location nouns is to use both the noun and adjective forms of the location and not just the adjective form as seemed to be suggested by the previous experiment. This reinforces the original premise that it is often the case that the location noun in the question is replaced in relevant documents by the adjectival form, as the experiments show that using the adjective boosts performance but cannot be relied upon for every question.

Showing that a particular approach to query formulation or expansion increases the coverage of the retrieved documents does not automatically imply that a QA system using these documents will show an increase in performance – higher coverage at the IR stage simply implies a higher upper bound on the performance of answer extraction components. To see if the increase in coverage, obtained via expanding the IR queries containing noun forms of locations with their adjectival form, has a beneficial effect on answer extraction components we provided the retrieved documents as input to the semantic type extraction system of Section 8.1.

The QA system was given as input the top 30 documents (the most significant results were observed when retrieving 30 documents, see Figure 7.5) and was evaluated using MRR (mean reciprocal rank, see (Voorhees, 2001)) over the top 5 answers returned for each question. The MRR of the QA system when given the documents retrieved using the question alone was 0.1947. Expanding the location nouns in these questions using the `alt` operator resulted in an MRR score of 0.1988. While there is an increase in performance of the answer extraction component it is not large enough to be statistically

significant although this could be due to the small number of questions used. Further evaluations over a larger test set are therefore required.

One possible explanation for the small increase in performance may be that while expanding the questions gives better coverage the answer bearing documents can now contain a word (the adjective form of the location noun) which is not part of the question. If the answer extraction components are not adapted to make use of this knowledge then they may well discard answers simply because they appear in a sentence which has little overlap with the original question.

7.2 Document Retrieval Over the World Wide Web

For QA systems that attempt to answer questions from closed collections, such as those evaluated at TREC, it is possible to experiment with different ways of improving the document or passage retrieval component, such as the approaches explored in the previous sections. Studies have been conducted to look at different passage retrieval approaches (Roberts and Gaizauskas, 2004; Tellex et al., 2003), multiple retrieval loops dependent upon the *quality* of the retrieved passages (Harabagiu et al., 2000), and building indexes containing syntactic or semantic knowledge (Fagan, 1987; Strzalkowski et al., 1996). While some of these approaches have proved more fruitful than others they all rely on having a reasonably sized closed collection over which they can operate.

With the continued growth in the amount of electronic text available via the Internet, more and more researchers are attempting to build QA systems that regard the Internet as a document collection from which to draw answers. For most researchers, indexing the Internet in an unusual fashion (indexing noun phrases for instance) is not even a remote possibility. QA systems designed to use the Internet have usually to rely on existing web search engines to retrieve a small set of documents ready for processing by an answer extraction component. Unfortunately, this means that although the question analysis component of such a system can generate a query tailored to a specific web search engine (Agichtein et al., 2001) the QA systems have to treat the IR system as a black box component (where they have no knowledge of the inner workings and see only the input and output from the IR system).

This leads to two questions which the remainder of this section attempts to address. Firstly, if one adopts a more relaxed view of QA evaluation than that used at TREC, and assumes that an answer is simply a short phrase or sentence which contains a supported instance of a correct answer, then are web search engines themselves adequate QA systems? Secondly, are the snippets returned by search engines suitable passages to be used by the more traditional QA systems? That is, can the techniques developed in the rest of this thesis be used to extract answers from the Internet as well as from closed collections such as AQUAINT.

The web search engines were evaluated using a small question set specifically constructed so as to avoid the TREC questions which now appear quite frequently on the Internet often

Search Engine	% Coverage at Rank									
	1	2	3	4	5	6	7	8	9	10
AllTheWeb	20	28	36	40	44	44	44	46	48	48
AltaVista	34	48	56	60	60	60	60	60	62	62
Google	28	48	56	58	64	64	64	66	70	72
Lycos	22	30	32	34	40	40	42	42	46	46

Table 7.2: Coverage results for the web search engines evaluated.

Search Engine	Answer Redundancy at Rank									
	1	2	3	4	5	6	7	8	9	10
AllTheWeb	0.20	0.40	0.56	0.76	0.82	0.92	0.98	1.10	1.20	1.34
AltaVista	0.34	0.56	0.86	1.12	1.32	1.58	1.82	2.04	2.16	2.36
Google	0.28	0.60	0.90	1.08	1.34	1.52	1.72	1.90	2.08	2.28
Lycos	0.22	0.36	0.48	0.62	0.70	0.78	0.90	1.02	1.18	1.28

Table 7.3: Answer redundancy results for the web search engines evaluated.

along with their correct answers. The question set is described in more detail in Appendix B with the following being a selection of example questions from the test set:

- *"Who was President of the USA from 1963 to 1969?"*
- *"What is the largest planet in our Solar System?"*
- *"How many stomachs does a cow have?"*
- *"How often does Haley's comet appear?"*
- *"Who performed the first human heart transplant?"*
- *"What is the normal colour of sulphur?"*

A web search engine is deemed to have correctly answered a question if a snippet describing a relevant document not only contains the correct answer but provides sufficient context to justify the answer. This is similar to the strict evaluation method (see Chapter 3) used throughout this thesis to evaluate systems answering TREC questions using the AQUAINT collection. By ensuring that the snippets, returned by the web search engines, justify their answers they can be treated as documents for the purposes of replacing traditional IR systems within the QA framework thus allowing existing QA techniques, such as those developed throughout this thesis, to be used over the web.

Four different web search engines were investigated; AllTheWeb, AltaVista, Google and Lycos. For each of these web search engines the questions were manually submitted to the search engines web interface and the top ten snippets were collected. Full coverage and answer redundancy results for the web search engines being evaluated can be seen in Table 7.2 and Table 7.3 respectively (note that this is a single informant evaluation performed by the author).

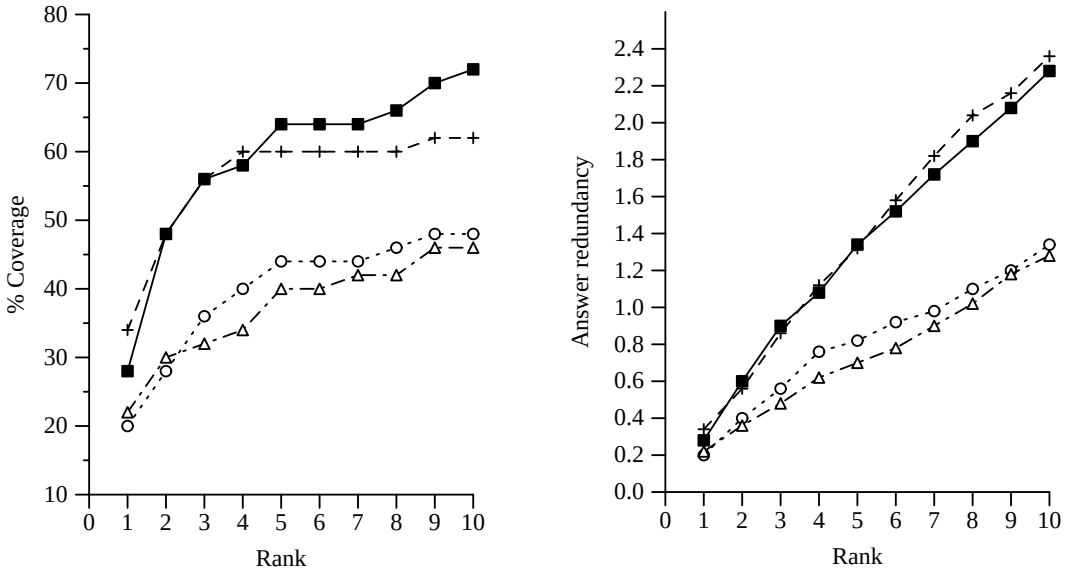

Figure 7.8: Coverage and redundancy results for AllTheWeb ○, AltaVista +, Google ■, and Lycos △.

These results are interesting for two reasons. Firstly, they show that if we consider coverage as being equivalent to the percentage of questions correctly answered at a given rank then certain web search engines (notably Google and AltaVista) are actually reasonably performing sentence based QA systems, especially when the first ten results are considered, with Google providing a correct supported answer to 72% of the questions. While it is difficult to compare these results with those of QA systems as most return exact answers rather than sentences we can compare the search engine results to the results reported at TREC-8 (Voorhees, 1999). TREC-8 included an evaluation of systems which were allowed to return up to 5 answers which could be up to 250 characters in length. Results at this task ranged from at least one correct answer to 11.6% of the questions (Eichmann and Srinivasan, 1999) up to the LASSO system (Moldovan et al., 1999) which returned at least one correct answer to 77.8% of the questions. Given that the questions used in TREC-8 were mainly reformulations of answer bearing documents and hence simple bag-of-words approaches were often sufficient for finding relevant sentences, the performance of the web search engines is comparable to that of the best TREC-8 system. At rank 5 Google returns at least one snippet which both contains the answer and justification for 64% of the 50 questions it was evaluated over while LASSO returned at least one answer bearing snippet (the snippet did not have to provide justification although it had to be with reference to a supporting document) for 77.8% of the TREC-8 questions.

The results in Table 7.2 also show that a QA system which uses a search engine as a passage retrieval system (i.e. uses just the returned snippets of text as input to an answer extraction component) cannot answer more than approximately 70% of the questions when relying on just the top ten snippets. Interestingly, Roberts and Gaizauskas (2004) showed that to achieve a similar coverage over the TREC questions and the AQUAINT collection (using Okapi as the search engine), systems had to consider the top thirty passages taken from the TREC collection. Similar results were reported by Hovy et al. (2000) in which they found that 8% of the questions still had no answers when the top one thousand segments were considered. From this it would seem that web search engines are capable of

acting as a suitable replacement for the document/passage retrieval components within a QA system.

The answer redundancy results for the web search engines, given in Figure 7.8, show that even the best performing search engines provide, on average, only two supported answer instances per question within the top ten snippets returned by the search engines. This suggests that QA systems may have to use the full documents to see a large number of answer instances (something which is known to improve the performance of QA systems (Light et al., 2001)), although Roberts and Gaizauskas (2004) showed that to achieve an answer redundancy of two or more, using the TREC questions and document collection, requires studying the top thirty passages and even considering the top two hundred passages does not result in an answer redundancy above five. This again reinforces the idea that the snippets produced by web search engines are appropriate passages for use in web based QA systems.

It should be noted that these evaluations say nothing about the performance of each search engine at locating relevant documents to sensibly worded IR queries, and only address how well they are able to produce answer bearing snippets from a naturally phrased question, i.e. these experiments evaluate search engines over a task they were never designed to perform.

7.3 Document Filtering

In Section 6.2.1 the notion of indexing documents based not only upon their words but also the semantic entities they contain was dismissed as infeasible especially when considering answering questions using the world wide web. While pre-indexing a collection may not be feasible it is possible to process documents as they are retrieved from the collection and discard those which do not contain entities of the expected answer type. In the previous experiments it was clear that using the top 20 documents retrieved from AQUAINT appeared to give the best end-to-end performance. At least some of these 20 documents will not contain entities of the correct type and so while they do not contribute noise to the retrieved documents they do reduce the chance of the documents containing at least one answer to the question.

The results given in Table 7.4 show the effect on coverage of retrieving the top x documents which contain an entity of the expected answer type, as well as the end-to-end performance when retrieving the top 20 documents containing the expected answer type. In fact a document is retrieved if it contains the expected answer type or one of its more general types as determined by the answer type hierarchy given in Figure 6.1. For example any answer to the question *"How tall is the Eiffel Tower?"* should be some form of distance measurement. However, when filtering the retrieved documents any document which contains either a distance, any measurement or an amount (a number with no known measurement units) will be accepted and passed to the answer extraction component. Table 7.4 gives the results not just over the TREC 2002 test set but also for the 33 distance, 4 time, and 7 temperature questions which were evaluated using the query

	Distance		Time		Temperature		TREC 2002	
	Std	**Filtered**	**Std**	**Filtered**	**Std**	**Filtered**	**Std**	**Filtered**
$c@1$	3.0%	6.1%	25.0%	25.0%	14.3%	14.3%	11.0%	17.5% ▲
$c@5$	18.2%	33.3% ▲	25.0%	50.0%	28.6%	28.6%	30.0%	40.8% ▲
$c@10$	27.3%	45.5% ▲	75.0%	100.0%	28.6%	28.6%	39.4%	49.1% ▲
$c@20$	45.5%	60.6% ▲	100.0%	100.0%	42.9%	42.9%	51.4%	57.9% ▲
$a@1$	12.1%	12.1%	25.0%	25.0%	14.3%	14.3%	21.0%	20.3%
$a@2$	21.2%	27.3%	50.0%	50.0%	14.3%	14.3%	26.1%	27.3%
$a@3$	27.3%	36.4% △	50.0%	50.0%	14.3%	14.3%	29.7%	31.5%
$a@4$	30.3%	36.4%	50.0%	50.0%	14.3%	14.3%	32.2%	35.0% △
$a@5$	33.3%	42.4% △	50.0%	50.0%	14.3%	14.3%	34.5%	36.7% △
$n@20$	85.5%	86.5%	64.6%	76.9%	85.7%	85.7%	84.4%	86.9%

Table 7.4: Results of filtering distance, time, temperature and all TREC 2002 questions.

expansion method detailed in Section 6.2.1 to allow easy comparison between the approaches.

Unlike the approach to improving the retrieval component, in Section 6.2.1, filtering the retrieved documents does not rely on expanding the IR query using keywords known to commonly occur with answer instances and so can be applied to all the questions for which the question analysis component can determine the expected answer type.

It is clear from these results that filtering the retrieved documents in this way has a significant effect on the coverage of the document set when evaluated over the complete TREC 2002 test set. Unfortunately while the IR results show significant improvement, this again does not result in a significant improvement in the accuracy of the answer extraction component. The accuracy figures do show improvement but only a small amount and so it is unclear if the extra processing required to filter the documents is worth the small improvement. Using Lucene to retrieve the top 20 documents from AQUAINT takes approximately 2 seconds per question[4]. Filtering the retrieved documents to retain the top 20 documents which contain answer type entities takes on average 40 seconds. A QA system which requires 40 seconds just to retrieve a handful of possibly relevant documents is probably of little use in real-time situations. It would be possible to reduce the time taken to filter the documents by pre-processing the collection and recording which documents contain instances of each semantic type but the fact that this is difficult was the original motivation for this post-retrieval filtering approach. This approach is especially useful for experimenting with document filtering as it requires little time or resources compared to constructing a semantic type index for a document collection.

7.4 Summary

This chapter has evaluated the performance of a number of novel ideas for document retrieval in a QA system (some of which were introduced in the previous chapter) as well as investigating known problems which effect the performance of IR engines and the

[4] When run on a PC with a 3GHz Pentium 4 CPU and 1 GB of RAM.

end-to-end performance of QA systems.

Others have investigated the ability of IR engines to retrieve relevant documents given a natural language question (Tellex et al., 2003; Roberts and Gaizauskas, 2004; Saggion et al., 2004) but often these studies have only been concerned with the performance of the IR engine. In this chapter we went further and investigated how increasing the coverage of a set of retrieved documents can affect the end-to-end performance of a QA system.

Figure 7.1 on page 75 shows that while the coverage continues to increase as more documents are retrieved the $a@1$ accuracy of the answer extraction component (discussed in the following chapter) peaks when only 20 passages, with a coverage of 51.35%, are considered. While it is difficult to generalise this finding to other IR engines and QA systems it is interesting to note that identical conclusions can be drawn when using the combination of Okapi and QA-LaSIE which suggests that it is important in all systems to balance approaches aimed at increasing coverage with their effects on end-to-end performance.

One possible reason for the decrease in accuracy as more documents are retrieved and coverage increases could be noise – the percentage of entities of the expected answer type which are incorrect answers to the question being asked. We introduced the $noise$ metric as a way of describing what happens to the search space as more documents are retrieved (see Section 3.2.3 for a formal definition). The experiments in this chapter have shown that increasing coverage leads to an increase in noise. This means that while the upper bound on the number of questions which can be answered is increased, the chance of selecting the correct answer is decreased.

In the previous chapter (page 68) we introduced a new ranking operator, called `alt`, as we believed it would be more appropriate for query expansion than the traditional `or` operator. Evaluations of using the `alt` operator for query expansion of both expected answer types (Section 7.1.2) and question words (Section 7.1.2) shows that the `alt` operator results in increases in coverage over the standard `or` operator. The small size of the question sets used in these experiments means that we cannot claim that the improvement is statistically significant in all cases but it certainly seems that the use of the `alt` operator should be investigated further.

Query expansion, independent of the ranking operator, was also investigated. In a similar way to Monz (2003) we performed query expansion based on the expected answer type of measurement questions, which while leading to an increase in coverage did not significantly improve the performance of a QA system. It is difficult to see how this approach could be expanded to cover other question types as in many instances there is not a small set of words which can be used to locate documents containing a given semantic type. Document filtering can, however, be used with all question types and was evaluated in Section 7.3. While the results of this evaluation show that filtering the documents so as to only consider those known to contain an entity of expected answer type significantly improves the coverage of the documents, we see no significant improvement in the end-to-end performance of a QA system. Of course it should be noted that just because these experiments show no end-to-end performance with one QA system does not mean they

will not benefit other systems. However, as a basis for the answer extraction systems to be introduced in the following chapter their application appears to be limited and so will not be used.

Finally, experiments in this chapter have shown that it should be possible to use the web as a source of documents using a search engine, such as Google, to retrieve a small selection of snippets. The snippets returned by Google have a coverage above that of documents retrieved from closed collections using IR engines such as Lucene. This will not only allow QA systems to be adapted to use the web but also means that such systems should be able to operate in real-time, as they will only have to process small snippets returned by the web search engines not long passages or documents.

Chapter 8

Answer Extraction

The final stage in a QA system, and arguably the most important, is to extract and present the answers to questions. This chapter documents two differing approaches to answer extraction: semantic type extraction and generalised surface matching text patterns.

8.1 Semantic Type Extraction

A naïve approach to extracting answers from documents would be to randomly choose a word or phrase and present this as an answer. While simple the system would be very unlikely to return correct answers to any questions.

A more principled approach is semantic type extraction which extracts all entities of a given type from the answer text and ranks them according to their frequency of occurrence within the relevant documents. In its simplest form semantic type extraction is only marginally more complex than randomly choosing a word or phrase. Given an answer type hierarchy, such as that detailed in Figure 6.1, semantic type extraction consists simply of extracting from text all entities of the expected answer type and then ranking them based on their frequency of occurrence within the relevant documents. Two answers can be considered equivalent if either they are identical, ignoring case differences, or the two answers match using the following similarity test (Brill et al., 2001):

> *Two answers are considered equivalent if and only if all the non-stopwords in one answer are present in the other or vice-versa.*

This similarity measure along with the answer type hierarchy in Figure 6.1 (remembering that this hierarchy is designed to cover all the semantic types the system can recognise in free text) and the question classifier of Section 6.1.1 can easily be used to develop a baseline semantic type answer extraction system. The evaluation of the answers returned by an answer extraction component involves not only the answers but the document provided in support of the answer. This is the strict evaluation regime discussed in detail in Chapter

	Baseline	S_2	S_3	S_4
$a@1$	17.56%	17.79% (+1.3%)	19.59% (+11.5%) △	20.94% (+19.2%) ▲
$a@2$	21.62%	22.07% (+2.1%)	25.00% (+15.6%) ▲	26.13% (+20.8%) ▲
$a@3$	24.55%	25.68% (+4.6%) △	28.15% (+14.7%) ▲	29.73% (+21.1%) ▲
$a@4$	26.58%	27.70% (+4.2%) △	29.73% (+11.9%) ▲	32.21% (+21.2%) ▲
$a@5$	27.48%	28.60% (+4.1%) △	30.63% (+11.5%) ▲	34.46% (+25.4%) ▲

Table 8.1: Accuracy of the semantic extraction system (improvements are with respect to the baseline).

3. For each answer returned by the baseline system the supporting document is the one in which the sentence containing the answer has the largest overlap with the question. If two sentences are equally likely to support the answer (i.e. the overlap between the question and the two sentences is equal) then the window containing both the answer and the question terms is determined and the sentence with the smallest window is deemed the most likely. The use of the smallest window was inspired by the idea of minimal span weighting (Monz, 2004) although here we are using minimal matching spans only to break ties in the simpler approach of word overlap.

The full accuracy results of using such a baseline system to extract answers from the top 20 relevant documents retrieved by Lucene for the TREC 2002 test questions can be seen in Table 8.1. These results show that even such a simple system is capable of answering a substantial portion of questions. In fact this baseline system returns correct answers at position 1 (i.e. $a@1$) for 17.56% of the questions. It is worth remembering at this point that the maximum accuracy the answer extraction component can achieve is less than 100% due to failings of the previous two components; the question analysis component may fail to determine the correct answer type and the document retrieval component may fail to select any answer bearing documents. For example using the top 20 documents limits an answer extraction component using a 100% accurate question analysis component to 51.35% (see Figure 7.2 on page 76).

8.1.1 Improved Answer Ranking

One approach to improving the baseline system is to improve the test used to determine if two answers are equivalent to each other. The baseline equivalence test is especially useful for questions requiring a person's name as an answer. For example the question *"Who is Tom Cruise married to?"* can be answered by both *"Kidman"* and *"Nicole Kidman"* and this equivalence test deems the two answers to be equivalent increasing the chance that this will be the top answer found by the system[1]. One of the problems with this approach, however, is its inherent naïvety especially when considering answers which contain numbers or dates. For example the two answers *"3,000 feet"* and *"three thousand feet"* should be considered equivalent as should *"March the 2nd 2004"* and *"02/03/2004"* but the baseline approach does not consider answers such as these to be equivalent.

We extend the baseline system to solve some of the more obvious mismatches between

[1] See Chapter 4 as to why these answers are correct even though Tom Cruise and Nicole Kidman are no longer married to each other.

Question	Tokenization Method	Answers From Articles Dated		Are Answers Equivalent
		4th March 2004	1st July 2004	
How far did he fall?	Normal	3,000 feet	three thousand feet	No
	Intelligent	3000 feet	3000 feet	Yes
When did he die?	Normal	yesterday	3rd of March	No
	Intelligent	3-Mar-2004	3-Mar-2004	Yes

Table 8.2: Answer comparison using normal and intelligent matching.

different realizations of the same information by introducing an intelligent system of tokenization. Consider the following two fragments of text describing a climbing accident:

> Written 4th March 2004:
> *...he fell 3,000 feet to his death yesterday...*
>
> Written 1st July 2004:
> *On the 3rd of March he fell three thousand feet to his death...*

Given these two excerpts the questions *"How far did he fall?"* and *"When did he die?"* could be asked. Assuming that the system can extract the answers to these questions from the text fragments it should be clear that they will not be considered equivalent given the similarity test mentioned above. The extracted answers, using a normal tokenization method, are shown in Table 8.2. If, however, when tokenising the answers to apply the similarity test we take a more intelligent approach and convert numbers and dates to a standard format then the similarity test will class the answers as equivalent to one another. The results of this intelligent tokenization approach can also be seen in Table 8.2. For both example questions this more intelligent approach to tokenization allows the system to recognise that the answers are equivalent and hence they will be ranked higher as the frequency of the combined answers is higher than either answer considered in isolation.

Note that while numbers are simply converted to a standard numerical format[2] dates are treated in a much more interesting fashion. Most electronic texts have an associated date, certainly newswire articles are linked to the date on which they were written, and this date can be used to resolve references to dates in the articles which on their own may not fully define a specific data. For example Table 8.3 shows a selection of text fragments which all resolve to 31-Aug-2004.

Extending the baseline's answer equivalence in this way gives system S_2, the results of which are shown in Table 8.1. These results show that the system performs better than the baseline system, significantly so at answer rank 3 or below. This is understandable considering that the difference between the systems means that multiple equivalent answers (which were not deemed equivalent by the baseline system) are now grouped together boosting the frequency of those answers resulting in the re-ranking of the list of answers produced for a question. While being a sensible approach this is unlikely to lead to dra-

[2] Parsing of numbers is carried out using ICU for Java from IBM.
`http://ibm.com/software/globalization/icu/`

Article Date	Text Fragment
30th August 2004	tomorrow
31st August 2004	today
1st September 2004	yesterday
–	Tuesday the 31st of August
–	August 31st, 2004
–	2004-08-31
Anytime in 2004	August 31st
Anytime in 2004	the 31st of August

Table 8.3: Different textual expressions of the same date.

matic improvements in $a@1$ over the full test set as it only alters the ranking of numeric or date based answers.

The baseline system, as has already been stated, works by extracting all the entities of the type assigned by the question analysis component from the documents retrieved by the document retrieval component. Given this, the only improvements possible are in the way in which the system groups and ranks answers[3]. The ideal approach to ranking and grouping entities into answers to a question would result in an accuracy for the end-to-end output of the QA system identical to the coverage of the documents retrieved by the document retrieval component for those questions which were correctly typed. From the results of system S_2 we can see that even if we consider $a@5$ the system is only capable of answering 28.60% of the questions (remember this is a lower bound on the accuracy of the system due to the strict evaluation metric, as described in Chapter 3, which requires not only a correct answer but an answer linked to a known supporting document).

While system S_2 has an improved grouping and ranking approach to the baseline it is still very simplistic giving $a@1$ of 17.79%. The ranking function used by both the baseline and S_2 is simply the frequency of occurrence of the answer within the documents. While this is a reasonable ranking algorithm, based upon the assumption that a correct answer will appear more frequently than other entities of the same type within answer bearing documents, it does not take into account any other link between the question and a given entity of the correct type appearing within the documents being processed.

In passages longer than a sentence in length there is no guarantee that the entity currently being considered as a possible answer to the question is in any way related to the words or phrases which caused the passage to be retrieved by the document retrieval component. It could well be that the entity being considered as an answer comes from a sentence which has nothing at all to do with the question. For instance in a narrative organised as a timeline each sentence may refer to a different moment in time, so for a question about when something occurred the answer will be in the same sentence as the event. Any dates in other sentences are completely irrelevant. For instance, document APW19990809.0178 entitled *Today in History* lists interesting events which took place on August the 10th. From this document a passage retrieval system could extract the following:

[3] Clearly improvements in both question classification and entity recognition would also improve the performance but we are assuming, for the sake of comparison, that the rules and resources used are fixed across the systems evaluated.

In 1809, Ecuador struck its first blow for independence from Spain.
In 1821, Missouri became the 24th state.
In 1874, Herbert Clark Hoover, the 31st president of the United States, was born in West Branch, Iowa.
In 1885, Leo Daft opened America's first commercially operated electric streetcar, in Baltimore.

If system S_2 was asked to answer the question *"When did Missouri become a state?"* using only the above passage then there are four entities of the expected answer type, any of which could in theory be the answer to the question. Clearly only 1821 is a correct answer to the question but system S_2 is unable to distinguish between the four dates. There are a number of possible approaches that could be taken to improve system S_2 to have a better understanding of the relationships between the entities of the correct type and the question including complex approaches such as logical reasoning about the semantics of the question and possible answer sentences (this kind of approach was mentioned in Chapter 5). Unfortunately, these approaches by their very nature tend to be time consuming and not necessarily suited to real-time question answering.

A simple approach which allows the answer extraction component to link the possible answer entities with the question involves taking into account the occurrence of the question words in the sentence containing the answer entity. Clearly the more question words (ignoring stopwords) that appear in a sentence containing an entity of the correct type the higher the likelihood that the entity is in fact the correct answer. This is the approach used to expand system S_2 to create system S_3. Given that a unique answer a to question q has occurred C_a times within the retrieved documents, with the instance used to determine the supported document occurring in sentence s, then the answer is scored using Equation 8.1:

$$score(a, q, s) = C_a \times \frac{|q \cap s|}{|q|} \tag{8.1}$$

The results of evaluating system S_3 in Table 8.1 are encouraging as they show a marked improvement over both the baseline and system S_2. Remember that no new answers can be found by this system only the ranking and selection of supporting documents can be improved and so even a small improvement is promising and an improvement of approximately 11% over the baseline system is statistically significant (see Section 3.3).

Whilst the end-to-end performance of QA systems is usually only concerned with the exact answer and a supporting document identifier, real world QA systems will need to supply simpler justification for each answer than a full document. One easy way, given the systems described in this section, would be to return the sentence used to determine the supporting document as justification for the answer. Unfortunately examining these sentences shows that in many cases sentences do not contain enough justification for the sentence to be unambiguous leaving it unclear whether or not the answer is actually correct. The problem is usually due to the fact that while the answer is in the sentence so are unresolved anaphora leaving it unclear if the sentence supports the answer when viewed in isolation. A clever approach to this problem would be to perform coreference resolution

to resolve the anaphora in the document and then as well as selecting the sentence containing the answer entity also select the sentences which contain the antecedents to any anaphora in the sentence which would hopefully result in the answer bearing sentence being unambiguous. However, as has been previously mentioned performing coreference resolution can be a time consuming task and so a simpler approach is called for.

The approach taken by the system S_4 is to calculate the overlap between the question and answer sentence as well as between the question and a short passage containing both the answer sentence and the preceding sentence, taking the longer justification if the overlap is higher. This approach assumes that anaphora can usually be resolved using the previous sentence (the same assumption taken by most coreference applications to reduce the search space, although Mitkov (1995) reports that antecedents have been found up to 17 sentences away from the corresponding anaphor) and the system only uses the previous sentence if the overlap is increased, which increases the chance of both the anaphor being resolved and of the justification being unambiguous. Of course this approach is not restricted to only extending the snippet so as to resolve simple anaphora as the following answer to the question *"What country was formed in 1948?"* (taken from document APW19980915.0380) illustrates:

> *When the state was formed in 1948, it had 806,000 residents.* Immigration has fuelled much of **Israel**'s population growth, with some 2.7 million people arriving in the nation's 50-year history.

Both system S_3 and S_4 return the second sentence containing the correct answer to the question, Israel. Only S_4, however, returns the first sentence which makes it clear that Israel is indeed the correct answer to the question. As well as providing better justification this affects the ranking of answers as the overlap between the supporting sentence (or snippet when expanded) s is part of Equation 8.1 which is used to rank the exact answers. The results of extending the approach to give system S_4 can be seen in Table 8.1.

It is clear from the results in Table 8.1 that the final semantic type extraction system significantly out performs the baseline system and shows a marked improvement over system S_3, which does not attempt to expand the justification text to resolve dangling anaphora, especially when the top 5 answers are considered. It should, however, be remembered that the evaluation of the systems only considers the answer in the context of the full supporting document and not the justification text extracted from the document. This means that while there is little difference in the extraction and ranking of exact answers (the task being evaluated) the improvements come in what is displayed to the user of the systems. In other words system S_4 is more likely to provide unambiguous justification of an exact answer than system S_3 while requiring little in the way of extra processing.

While the S_4 answer extraction component only achieves an $a@1$ of 20.94% it is important to remember that users of real-world QA systems would probably be happy with a system which returned 5 or 10 answers along with justification. Most users of web search engines prefer a relevant document within the first page of results, which for most

search engines means 10 documents. Considering the $a@5$ performance 34.36% of S_4 is therefore a fairer measure of the use of this component within a real-world QA system. We should remember of course that the answer extraction component is limited by the performance of the document retrieval component. In these experiments the coverage of the documents passed to the answer extraction component, and hence the upper bound on the performance of S_4, is 51.35% (see Figure 7.2 on page 76). Given this the answer extraction component returns a correct exact answer at position 1 for approximately 40% of the test questions that it can be expected to answer and returns a correct answer within the first five answers for approximately 67% of the questions. These results tell us two important things. Firstly that given good input (i.e. answer bearing documents) this relatively straight forward method of answering questions performs well. Secondly that a good document retrieval component is vital for good end-to-end QA performance.

8.1.2 Dynamically Expanding the Answer Type Hierarchy

The main problem with this semantic type extraction approach to question answering is the number of questions which are assigned an UNKNOWN type, i.e. questions for which the system cannot determine the expected answer type and so does not attempt to answer. Clearly the answer type hierarchy could be extended to cover new answer types as detection systems are developed for new semantic categories. As an intermediate strategy, however, it is possible to dynamically modify the answer type hierarchy using WordNet to increase the questions which can be successfully classified and possibly answered.

This addition to the system works by attempting to determine the semantic type of the word or phrase in a question which specifies the type of entity being sought. For example the question *"What grapes are used in making wine?"* is clearly asking for the names of grape varieties. Assuming that the system can correctly extract *grape* from the question then the system can insert grape as a top level entry in answer type hierarchy. As was stated in Chapter 6 the answer type hierarchy is constructed so as to cover only those types which the answer extraction component can locate in free text. Given this the system needs a way to locate, in this instance, grape varieties in free text. This extension to the semantic type extraction approach uses WordNet to construct a list of all known grape varieties by extracting the hyponyms of WordNet's grape entries. This list can then be used to tag all the grape varieties in the texts retrieved by the previous component of the QA system.

When combined with the standard answer type hierarchy of Figure 6.1 this extension adds only five new nodes to the hierarchy for the questions from TREC 2002. Of these five questions WordNet contains correct answers to two of them both of which are correctly answered by the QA system (although one answer is given as unsupported because the supporting document was not previously known to contain a correct answer to the question). For example the question *"What galaxy is closest to the milky way?"* was not assigned a type by the question classification scheme detailed in Section 6.1.1 but is suitable for answering via WordNet. The system, using a set of 11 hand crafted regular expressions, can correctly determine that in this example the answer should be a galaxy.

Using WordNet to determine possible answers of type *galaxy* gives; *spiral galaxy, spiral nebula, Andromeda galaxy, Great Attractor, Magellanic Cloud, Large Magellanic Cloud* and *Small Magellanic Cloud*. Any occurrence of these entities within the retrieved documents is then tagged as type *galaxy* allowing the semantic type extraction system to proceed as normal and correctly respond to the question with the answer *Andromeda galaxy*.

Interestingly a simple approach to answering multiple choice questions could be implemented along similar lines. Instead of using WordNet to determine the list of possible answers, the answers provided with the question could be used to tag retrieved documents. Answers could then be extracted and ranked in the same manner as with the answers to any question found using the standard semantic extraction approach.

8.2 Generalised Surface Matching Text Patterns

One of the main issues surrounding the semantic type extraction approach to answer extraction which the final system presented in Section 8.1 attempted to solve was that no specific link is made between entities of the correct type and the question being asked. This section presents an approach to answer extraction which attempts to learn patterns describing how correct answers to specific question types are related to the main question term. Patterns are acquired from the surface structure of text and do not include syntactic or deep semantic information.

The most significant component of this approach to answer extraction is the acquisition of surface matching text patterns using a set of question-answer pairs. The following sections describe how such patterns can be acquired and how they can then be used to find possible answers to previously unseen questions. It should be noted that this approach was motivated by the surprising performance of a similar system reported by Soubbotin and Soubbotin (2001) as well as the need to extend a specific implementation (Ravichandran and Hovy, 2002) to handle more complex answer realisations.

8.2.1 Learning Pattern Sets

A different pattern set is required for each different question type the system is required to answer. Learning generalised surface matching text patterns is a two stage process of acquisition and analysis. The easiest way to describe both stages of the process is though an example. For easy comparison with the work of other researchers (Soubbotin and Soubbotin, 2001; Ravichandran and Hovy, 2002) the example *"When was X born?"* is used. For this example the acquisition algorithm works as follows:

1. A collection of twenty example questions, of the correct type, and their associated answers is assembled.
2. For each example question a pair consisting of the question and answer terms is

produced. For example “Abraham Lincoln” – “1809”.

3. For each example the question and answer terms are submitted to Google, as a single query, and the top 10 documents are downloaded[4].
4. Each retrieved document then has the question term replaced by the single token AnCHoR.
5. Depending upon the question type other replacements are then made. In this example all dates, locations, organisations and person names are replaced by representative tags DatE, LocatioN, OrganizatioN and PersoN. For other question types differing entities may be replaced by tags. If any of these tags replace text which contains the answer term then a compound tag, such as AnSWeRDatE is assigned.
6. Any remaining instances of the answer term are then replaced by AnSWeR.
7. Sentence boundaries are determined and those sentences which contain both AnCHoR and AnSWeR are retained.
8. A suffix tree (Ukkonen, 1995) is constructed using the retained sentences and all repeated substrings containing both AnCHoR and AnSWeR and which do not span a sentence boundary are extracted.

This produces a set of patterns, which are specific to the question type. Continuing with the example of *“When was X born?”* a selection of patterns produced by this process are:

```
from AnCHoR ( AnSWeRDatE - DatE )
AnCHoR , AnSWeRDatE -
- AnCHoR ( AnSWeRDatE
from AnCHoR ( AnSWeRDatE -
```

Unfortunately these patterns contain no information about how accurate they are when used to answer unseen questions, so the second stage analyses the patterns, removing those which are of little value in answering unseen questions. The analysis algorithm is then as follows:

1. A second set of twenty question-answer pairs are collected and each question is submitted to Google and the top ten documents are downloaded.
2. Within each document the question term is replaced by AnCHoR.
3. The same replacements as carried out in step 5 of the acquisition phase are also performed on the document in this stage and a table is constructed of the inserted tags and the text they replace.

[4] The documents are actually downloaded from Google’s cache to guarantee that the version of the page indexed by Google and not a more recent revision is used.

Pattern	Precision
AnCHoR \((DatE) - DatE \)	0.909
AnCHoR \((DatE) -	0.909
AnCHoR \((DatE)	0.738
AnCHoR (DatE) - DatE	0.700

Table 8.4: Analysed patterns for questions of the form *"When was X born?"*.

4. Each of the previously generated patterns is converted to a standard regular expression designed to capture the answer text, giving expressions such as:

```
from AnCHoR \( (DatE) - DatE \)
AnCHoR , (DatE) -
- AnCHoR \( (DatE)
from AnCHoR \( (DatE) -
:  AnCHoR , (DatE) -
```

   These expressions allow us to easily retrieve the single token which `AnSWeR` (or one of its extended forms such as `AnSWeRDatE`), in the original pattern would have matched against.

5. Each of the previously generated patterns is then matched against each sentence containing the `AnCHoR` tag. Along with each pattern, P, two counts are maintained: C_a^P, which counts the total number of times the pattern has matched against the text and C_c^P, which counts the number of matches which had the correct answer or a tag which expanded to the correct answer as the text extracted by the pattern.

6. After a pattern, P, has been matched against all the sentences if C_c^P is less than five it is discarded. The remaining patterns are assigned a precision score calculated as C_c^P/C_a^P. If the pattern's precision is less than or equal to 0.1 then it is also discarded[5].

Using this method to produce analysed patterns for the question type *"When was X born?"* gives patterns such as those in Table 8.4, which can now be used to locate answers to unseen questions.

This approach to acquiring patterns has been used to produce pattern sets to answer the following types of question: *What is the abbreviation for X?*; *When was X born?*; *What is the capital of X?*; *What country is X the capital of?*; *When did X die?*; *What does X stand for?*

In all of these pattern sets the names of people and organizations as well as dates and locations are replaced by associated tags. The pattern set for questions of the form *"What does X stand for?"* also involves replacing noun chunks with the tag `NounChunK`.

This implementation differs from those previously reported (Soubbotin and Soubbotin, 2001; Ravichandran and Hovy, 2002) in that it specifically addresses the problem of

[5] These cut-off values were adopted based on empirical observations made during development.

overly specific patterns (we originally discussed this in (Greenwood and Gaizauskas, 2003)). The problem with the original approach was that only the question and answer terms were generalised by being replaced by representative tags. This meant that for question types were it was likely that question specific words could appear between the question and answer term it would be difficult to acquire useful extraction patterns. More generally any acquired pattern must consist of three components 1) the AnCHoR tag (which gets initialised as the question-specific anchor), 2) the AnSWeR regular expression, and 3) literal text occurring between 1) and 2). In the original descriptions of this approach component 3) could not be generalised, i.e. could not be a regular expression containing meta-characters, and hence it could only match itself. So while generalised patterns could be acquired to extract the date of birth from *"Mozart (1756-1791) was a musical genius"* it was not possible to acquire a general pattern which could extract the date of death due to the specific date of birth appearing between Mozart and 1791. By extending the basic approach to give the algorithm detailed in this section we allow not only the question and answer terms to be generalised but also other semantic entities such as dates which allow general patterns to be acquired for a larger range of questions.

The relatively small number of pattern sets produced using this method is not due to any failings or shortcomings of the approach but is solely due to the time required to generate the training data. Each pattern set requires data to be collected for 40 separate examples (plus examples for evaluation purposes). Generating these examples is time consuming and limits the applicability of this approach.

8.2.2 Pattern-Based Answer Extraction

One of the problems with the use of pattern sets for QA is determining which set to use to answer a given question. The answer type hierarchy of Figure 6.1 does not contain nodes for specific question types but rather for answer types. This means, for instance, that while date is a member of the hierarchy, date of birth is not. It would not in fact make sense to include specific question types in the hierarchy as this would result in an extremely large and unwieldy hierarchy. A simpler approach is to define a set of rules (in this instance regular expressions) that both match against questions which a given pattern set can answer and can extract the AnCHoR term from the questions as required by the approach. If the selected pattern set cannot locate any answers then the standard answer type hierarchy can be employed to determine the expected answer type and extract entities of the correct type. For example with questions asking when someone was born, rules which match and extract the AnCHoR from the following formulations are required:

- *When was AnCHoR born?*
- *What year was AnCHoR born?*
- *What date was AnCHoR born?*
- *In which year was AnCHoR born?*
- *AnCHoR was born in which year?*

Pattern Set	% Correct	% Wrong	% Unanswered
What is the abbreviation for X?	78	7	15
When was X born?	60	8	32
What is the capital of X?	29	61	10
What country is X the capital of?	55	40	5
When did X die?	54	1	45
What does X stand for?	21	30	49
Average	**49.5%**	**24.5%**	**26.0%**

Table 8.5: Evaluation of the individual pattern sets.

- *On what date was* AnCHoR *born?*

The matching rules associated with each acquired pattern set are applied to unseen questions, in no particular order, until a match is found or until all the sets have been exhausted. If the question is matched by a rule then the AnCHoR is extracted ready for use in answering the question. Using a pattern set to find answers to a question is extremely simple. Each document retrieved by the IR component of the QA system is pre-processed by replacing the question term by AnCHoR and then any other associated replacements are made (the same as in step 5 of the acquisition phase).

Each pattern within the set is then matched against the sentences containing AnCHoR and for each successful match the token captured by the expression (or if the token is a tag then the text the tag replaced) is stored along with the precision of the pattern. When all the patterns have been applied to all the sentences any answers which have been located are grouped together, using the same approach as employed to group answers found using the semantic type extraction method (see Section 8.1). The answer groups are then ranked based upon the accuracy of the best acquired pattern that located the answers within the group.

8.2.3 Pattern Set Accuracy

Each analysed pattern set was evaluated over one hundred unseen examples with relevant documents being downloaded from the Internet via Google. The results of this evaluation, which can be seen in Table 8.5, show that although some of the pattern sets perform well returning very few wrong answers others, such as those for answering *"What does X stand for?"*, perform poorly.

One of the reasons for the poor performance of some pattern sets is the number of very general rules which are acquired by this method. For instance the pattern set acquired for questions of the form *"What is the capital of X?"* include the very general pattern: of AnCHoR, AnSWeR. Clearly such a pattern will match against a large number of sentences not all of which will extract a country or state or even a more general location instance. This is illustrated by the fact that the precision of this pattern is only 18%. While one approach would be to insist that the text extracted as answer had the correct semantic type this is not always feasible. For instance what is the semantic type of an abbreviation? How would a system know that NASA is an abbreviation and not a normal word? It

may be that using more example question-answer pairs during the learning process would produce more high quality patterns which would be used first when answering questions.

The main drawback to this approach, and the reason that currently only six pattern sets have been acquired, is the number of question-answer pairs required for the learning method to acquire reasonable surface matching patterns. Requiring a large number of example question-answer pairs as training data poses two problems. Firstly collecting the training data is a time consuming task. For some question types lists or databases populated with the question-answer pairs may already be available (for instance there are many web sites that list the dates of birth and death for famous people which can be easily mined including `http://www.famousbirthdays.com`) but for most question types the question-answer pairs will have to be manually collected. The second problem limiting the more widespread use of surface matching text patterns is that many relevant documents are required for each question-answer pair and this usually means that closed collections are not suitable for use with the learning algorithm. Instead the patterns have to be acquired from the web – for almost every question-answer pair imaginable there will be at least one relevant page on the Internet. This is an issue as the writing style can differ dramatically from general interest pages available on the Internet to the newswire articles usually found in closed collections. This means that while some of the acquired pattern sets perform well when used to answer questions from the web they perform poorly when used against collections such as AQUAINT. For instance while the average accuracy of the six pattern sets shown in Table 8.5 is 49.5% this drops to only 6.1% (2/33) when the same sets are used to answer appropriate questions from the TREC 2002 test set.

The pattern acquisition method detailed in Section 8.2.1 uses question-answer pairs to learn the surface text which often links them together. As has already been mentioned this requires a large amount of hand collected training data (the question-answer pairs) making it a time consuming and expensive approach. The alternative is to attempt to learn new patterns starting from a pattern known to be useful for the task. For example, Yangarber et al. (2000) present an unsupervised method for the discovery of information extraction (IE) patterns. This approach allows the discovery of subject-verb-object (SVO) patterns which are relevant to a specific IR task, such as management succession. Starting from a small set of seed patterns, such as `COMPANY-appoint-PERSON`, the method acquires other patterns which commonly co-occur with the seed patterns and hence are likely to also be relevant. While this does not lead to a complete IE system (without manual intervention) it does allow relevant documents within a corpus to be identified. In a similar way Stevenson and Greenwood (2005) present an approach to discovering SVO patterns based on semantic similarity to the seed patterns. These methods could allow a system to learn patterns for specific answer types from a small handful of examples. For example the question type *"Who discovered `LOCATION`?"* can be represented using the subject-verb-object triple `PERSON-discover-LOCATION`. One of the previously mentioned algorithms could then be used to determine other similar patterns which also encode answers to the question, such as `PERSON-discover-CHEMICAL_ELEMENT`. Clearly these approaches would have to be extended beyond the SVO representation (see Sudo et al. (2003) for one possible way of extending the representation) to allow question types such as *"When was X born?"* to be answered using patterns learnt in such a way.

8.3 Answer Extraction Limitations

Ignoring errors introduced by previous components of a QA system (incorrect question analysis or no relevant documents retrieved) there are a number of limitations common to both answer extraction components described in this chapter.

8.3.1 Semantic Type Recognition

Both approaches to answer extraction described in this chapter rely on semantic type recognition. The semantic type extraction component can only extract answers that have been recognised in free text and so the performance of the extraction component is directly linked to the performance of the semantic type recogniser. The surface matching text patterns may contain generalised labels which match against semantic entities recognised in the text (for example the patterns may contain `DatE` labels which match dates in the relevant documents).

Whilst the semantic type recogniser is therefore of critical importance to the performance of the answer extraction components it is difficult to determine the precision of the semantic type recogniser used in these experiments. While accepted evaluation sets are available for named entity recognition they usually only evaluate a small number of types. For instance the MUC named entity evaluations (Chinchor, 1998) only consider: person, location, organization, date, time, money, and percent.

The semantic type recogniser used throughout this thesis is based upon the gazetteer and named entity (NE) components of the ANNIE system distributed with the GATE framework (Cunningham et al., 2002). The ANNIE NE system achieves an F-measure of approximately 90% on newswire texts (Maynard et al., 2003), for the MUC semantic types. This is a good basis for the extended system, which covers all the semantic types in the answer type hierarchy on page 55, and which is used by the semantic type extraction component of Section 8.1.

While no performance figures are given for the extended NE system, it should be noted that the additions to the basic ANNIE system were made with care so as not to degrade the performance of the system over the MUC semantic types.

In essence if the semantic type recogniser is unable to recognise an entity in free text then the answer extraction components will not be able to use the entity (either as an answer or as context in a surface matching text pattern) and hence the performance of the answer extraction components is likely to suffer.

8.3.2 Questions With No Known Answers

It is possible that a question could be asked for which the document collection being used does not contain a correct answer. This can happen frequently with closed document

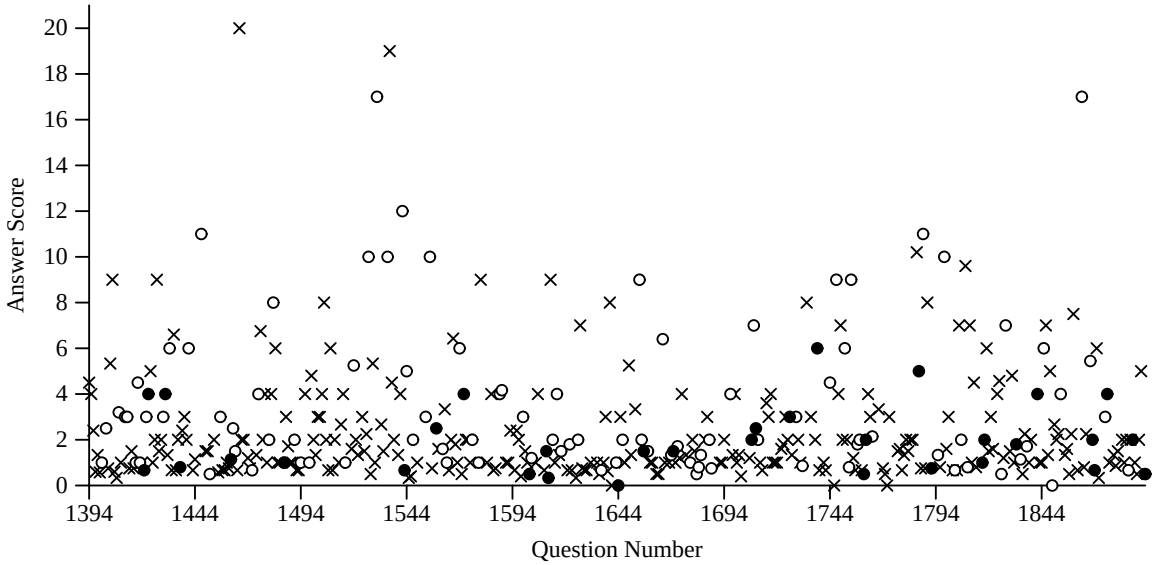

Figure 8.1: Score for the first returned answer (Correct answers ○, wrong answers ×, and wrong answers to questions with no known answer ●)

collections, such as AQUAINT. Questions can be specifically engineered to have no answer in a closed document collection by asking about an event occurring after the date of the last document in the collection (for the AQUAINT collection any question concerning events occurring after the 30th of September 2000 will be guaranteed not to have an answer in the collection). Neither of the approaches to answer extraction detailed in this chapter explicitly handles such questions. While the approaches can fail to find an answer to a question (and hence return NIL) they cannot, having found a set of possible answers, determine that none of them are correct.

For example, the semantic type extraction component detailed at the beginning of this chapter scores answers using Equation 8.1 and plotting the answer score for the first answer returned for all the questions used in the TREC 2002 QA evaluations gives Figure 8.1. This question set contains 32 questions for which there is no known answer in the AQUAINT collection but for which the answer extraction component returned a non-NIL answer. While it would appear from Figure 8.1 that answers returned for questions without a known answer have relatively low scores they are within the same range as the scores for the majority of correct answers to questions. This suggests that while the answer scoring function may be able to rank a correct answer above incorrect answers to the same question it would be inadvisable to compare the scores of answers across different questions in an attempt to determine if a question has no correct answer.

8.4 Summary

This chapter has introduced two approaches to answer extraction: semantic type extraction and surface matching text patterns.

Our approach to semantic type extraction works by extracting all entities of the expected

answer type from the retrieved documents. The extracted answers are then grouped to remove duplicate answers and a ranking algorithm is then applied to select the most likely answer. The expected answer type is determined using the manually constructed question classifier introduced in Section 6.1. This means that only questions for which the answer type falls within the hierarchy on page 55 can be answered. We introduced a number of techniques for determining the similarity between answers including converting numbers and dates to a common format for more accurate comparison of candidate answers. Candidate answers are then ranked based on their frequency of occurrence within the retrieved documents and the overlap between the question and the sentence the answer appears in. This gives a ranking algorithm which uses two features previously shown to be good guides to answer selection (Light et al., 2001). This approach allows the system to return a single correct answer to approximately 20% of the questions (this increases to approximately 35% if the first five answers are considered).

The surface matching text patterns were inspired by earlier approaches (Soubbotin and Soubbotin, 2001; Ravichandran and Hovy, 2002), which while showing promise appeared to suffer from a lack of generality. That is, question-specific details could remain in the acquired patterns reducing their applicability to unseen questions. Our approach avoids this problem by allowing a number of semantic generalisations during pattern acquisition. While the patterns can be effective at retrieving correct answers (a correct answers is returned for 78% of the *"What is the abbreviation for X"?* style questions) they suffer from two problems that limit their applicability. Firstly they require a large amount of training data to produce and secondly their performance appears to degrade quite significantly if they are used over text of a different style than that from which they were acquired. The main benefit of this approach is that when a pattern selects an answer, in most cases that answer is likely to be correct (average precision of approximately 60%) although there are many questions for which no answer is found (average recall of approximately 50%).

Combined these two approaches allow a large number of factoid questions to be attempted and an above average number to be correctly answered (see Appendix E). This is because the surface matching text patterns can be used first (high precision but low recall) and if they fail to find an answer then the semantic extraction component can be used instead. This is in fact the approach employed in AnswerFinder, a web based QA system introduced in the following chapter.

It should be noted that both the semantic type extraction and surface matching text pattern approaches to QA are similar in idea to semantic annotation, i.e. assigning semantic category labels to a wider range of phenomena than just named entities. For example, Guthrie et al. (2003) developed a number of approaches to semantic annotation, with a coarse (25 category) hierarchy of types derived from LDOCE (Procter, 1978), in order to overcome the data sparseness problem associated with many IE tasks. Their aim was similar to the surface matching text patterns in that using semantic annotations allows both the sentences *"The IRA bombed a family owned shop in Belfast yesterday"* and *"FMLN set off a series of explosions in central Bogota today"* to be represented as `Organization Attacked Location Date` reducing the data sparseness problem and hence providing more training data for tasks such as the acquisition of information extraction rules.

The main difference between their approach and ours is that whilst they use high level annotations allowing the labelling of the majority of nouns in text we use narrow categories which cover only a small set of entities. Categories which cover only a small set of very similar entities are extremely useful in narrowing the search space for question answering but tend to result in a large percentage of entities having no annotation.

Chapter 9

Factoid QA with AnswerFinder

AnswerFinder was originally developed as an interface to an early version of the semantic type extraction system detailed in Section 8.1 simply returning as answers, any entities of the expected type (e.g. for a question such as *"Where is ..."* all locations in the relevant document would be considered as possible answers). The aim of this system was to introduce the millions of Internet users to the benefits of question answering technology (Greenwood, 2004a). Unfortunately there are a wide variety of questions which this system could never answer as it simply did not know how to approach them (see Section 8.1).

9.1 Underlying QA Technology

The current version of AnswerFinder[1] is not a question answering system per se but rather a framework in which multiple question answering systems can be hosted. The standard distribution of AnswerFinder currently contains three such question answering systems:

- the surface matching text patterns detailed in Section 8.2, and
- the semantic type extraction system detailed in Section 8.1,
- a naïve system to answer simple translation questions such as *"What is the Spanish for 'pig'?"* or *"How do you say cat in French?"*. This system works by extracting the word or phrase and the language from the question using a set of regular expression, translating it using AltaVista's Babel Fish translation service[2] and then looking for documents which contain the word or phrase in both languages.

Together these three systems answer a wider range of questions than the original implementation and as the question answering API is freely available users can now easily

[1] Freely available from `http://www.dcs.shef.ac.uk/~mark/phd/software/`

[2] `http://babelfish.altavista.com`

develop their own question answering systems to further extend the range of questions which AnswerFinder can answer.

The order in which the QA systems are used can be altered by the user but by default they are consulted in the order given above. This ordering is used because if the surface matching text patterns find an answer (not guaranteed) it is likely to be correct while the semantic type extraction system is likely to always find an answer but with less certainty that it will be correct. The hosted QA systems are consulted in turn to see if they are able to process a given question. Once a system states that it can answer a question then it is asked to return a list of answers using a specified search engine to locate relevant documents[3]. Currently relevant documents (or snippets if the user prefers) are located and downloaded from the web using the Google Web API.

As each QA system can return multiple answers to a question, and AnswerFinder can be configured to display between one and ten answers to a question, it would be useful to also display the level of confidence the system has in each of the proposed answers. While this is calculated by the separate QA systems hosted inside AnswerFinder, we will briefly explain how it is computed for the answer extraction components detailed in Chapter 8.

For answers produced by the surface matching text patterns the confidence assigned to each answer is simply the highest precision (converted to a percentage) of the patterns which located the answer. The precision of a surface matching text pattern is determined using twenty questions (not used to acquire the patterns) during analyse of the pattern prior to its use. Pattern analysis is discussed in Section 8.2.1. The precision of a pattern is an accurate measure of the ability of the pattern to select a correct answer to a given question and is therefore a suitable confidence measure.

Calculating a confidence level for the semantic type extraction system of Section 8.1 is more complex than for the surface matching text patterns. Whilst the equation (Equation 8.1) used to score answers is an effective ranking measure for answers to a single question (see discussion in Section 8.3.2) it is unable to distinguish correct answers across multiple questions. As such it cannot be used directly as a confidence measure. The best answers are assumed to be those which not only occur frequently in the relevant documents but do so in at least one sentence which has a large overlap with the question. The score of an answer a can therefore be mapped to a percentage confidence using Equation 9.1 in which F is the frequency of the most frequently occurring answer for the question and $score(a)$ is the score of answer a as given by Equation 8.1.

$$confidence(a) = \frac{score(a)}{F} \times 100 \tag{9.1}$$

This equation guarantees that not only will the confidence level decrease as more answers are considered but it also reflects both the frequency and overlap components of the answer scoring function. The first answer will only have a confidence of 100% if it is both the most frequently occurring answer and its supporting snippet overlaps 100% with the question.

[3] For further technical details see `http://www.dcs.shef.ac.uk/~mark/phd/software/`

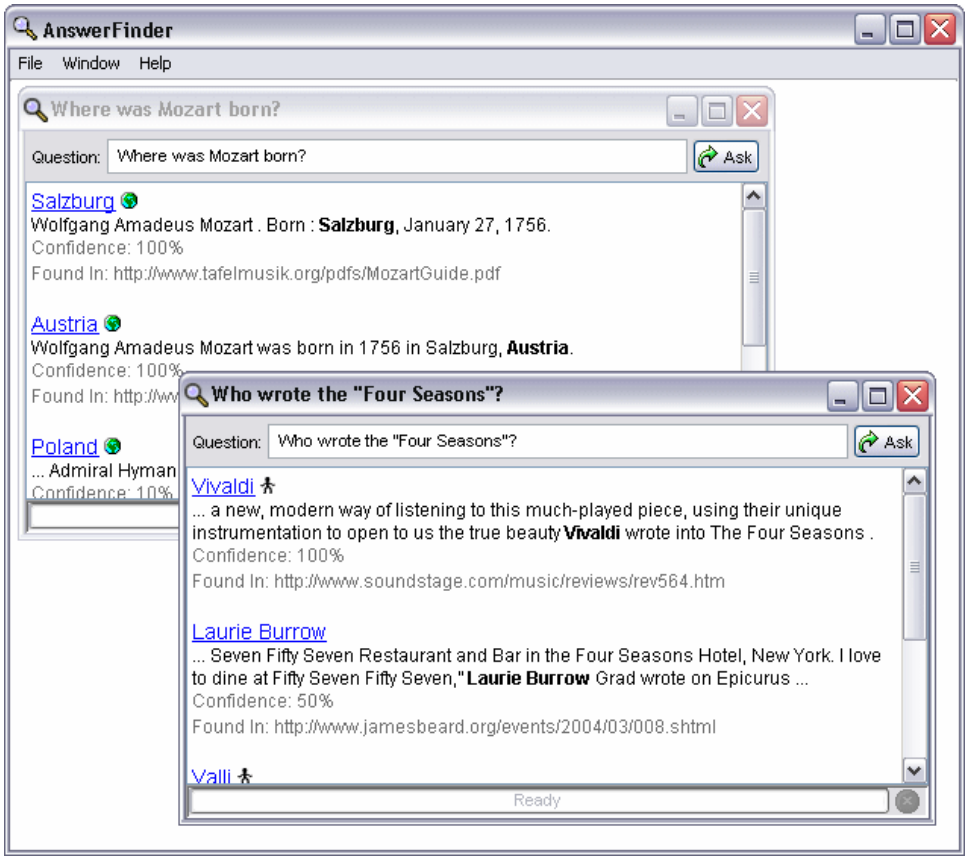

Figure 9.1: AnswerFinder: an open-domain factoid question answering system.

9.2 User Interface

AnswerFinder's user interface, which can be seen in Figure 9.1, was designed to make it as easy as possible for an average computer user, familiar with web browsers, to make use of the question answering technology detailed in Part II of this thesis. The application uses a multi-document interface to allow the user to ask numerous questions at the same time and each window consists of an area for the user to type their question and an area in which to display any possible answers.

As well as providing both the isolated answer and the context in which it occurs AnswerFinder will, where appropriate and available, provide links to extra resources that expand upon the answers given to the question, this currently includes:

- Biographies of the people listed as answers to a question.
  `http://www.infoplease.com/people.html`
- Maps of the locations given as answers to a question.
  `http://www.multimap.com`
- Weather forecasts for the locations listed as answers to a question
  `http://weather.yahoo.com`

While it would be preferable to display these resources within AnswerFinder along with the associated answers this is not currently possible due to the terms of service of the websites from which the information is collected.

9.3 Comparisons with other Web Based QA Systems

The idea of building an easily accessible question answering system which uses the web as a document collection is not new. Unfortunately it is difficult to determine the first system of this kind due mainly to the fact that the authors of many systems claim to have been the first to develop and make public such a system. All of these systems are accessed via a web browser and unlike AnswerFinder involve no client-side software (although client-side software is not an issue if the popularity of Google's toolbar is any indication). Their aim, however, is the same – to go beyond standard document retrieval. In the remainder of this section we will compare a number of these systems to AnswerFinder.

The consistently best performing system at TREC (Moldovan et al., 1999; Harabagiu et al., 2000; Harabagiu et al., 2001; Moldovan et al., 2002) forms the backbone of the PowerAnswer system from Language Computer[4]. From a user's point of view the system is similar to AnswerFinder in that the full question is given to the system and then answers are displayed. The difference is that the answers are very much what you would expect from a search engine in that each answer is a sentence and no attempt is made to cluster (or remove) sentences which contain the same answer. This means that the user still has to read the sentences to locate the answer to their question. This is strange given that fact that at TREC the underlying technology has been shown to be highly accurate (approximately 85%) even when returning only a single exact answer (Moldovan et al., 2002).

A system called AnswerBus[5] (Zheng, 2002) behaves in much the same way as PowerAnswer, returning full sentences containing duplicated answers. The reason for mentioning it here is that the questions can be given to the system in either English, French, Spanish, German, Italian or Portuguese with the system automatically determining the language, although answers are only given in English. It is claimed that AnswerBus can correctly answer 70.5% of the TREC 8 question set although we believe the performance would decrease if exact answers were being evaluated as experience of the TREC evaluations has shown this to be a harder task than locating answer bearing sentences.

Much closer to AnswerFinder is a system called NSIR[6] from the University of Michigan. NSIR uses a standard search engine to locate relevant documents, just as AnswerFinder does, and returns ranked exact answers. Unfortunately no context is provided along with the answers so a user still has to read the original document to verify that a given answer is correct. The system was entered into the TREC 2002 evaluation (Qi et al., 2002) and correctly answered 24.2% of the questions (this includes those marked as inexact or not

[4] `http://www.languagecomputer.com/demos/`

[5] `http://misshoover.si.umich.edu/~zzheng/qa-new/`

[6] `http://tangra.si.umich.edu/clair/NSIR/NSIR.cgi`

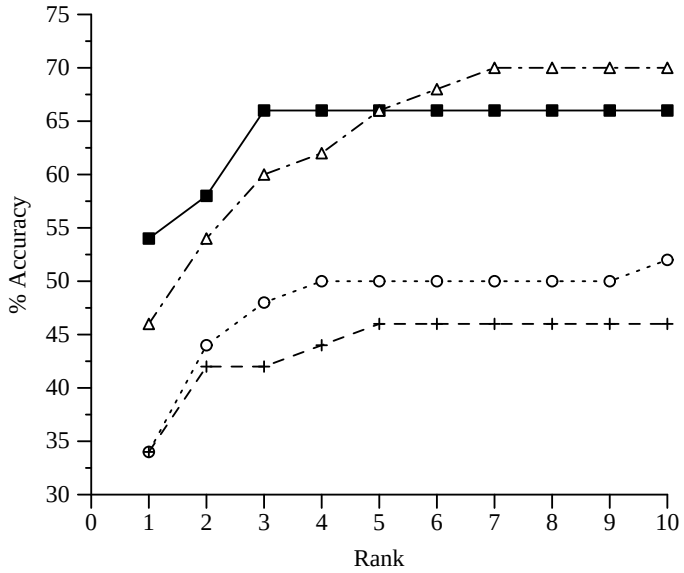

Figure 9.2: Comparison of AnswerBus ○, AnswerFinder ■, IONAUT +, and PowerAnswer △.

supported) which is comparable to the 25.6% obtained by AnswerFinder over the same test set.

The system most comparable with AnswerFinder, from a user's perspective, as it accepts unstructured natural language questions and returns exact answers and supporting snippets is IONAUT[7] (Abney et al., 2000). IONAUT uses its own crawler to index the web with specific focus on entities and the relationships between them in order to provide a richer base for answering questions than the unstructured documents returned by standard search engines. The system returns both exact answers and snippets. Unfortunately the exact answers are not tied to a specific snippet, so it is not immediately clear which snippet supports which answer. This problem is compounded by the fact that multiple snippets may support a single answer as no attempt has been made to cluster/remove snippets supporting the same answer.

We believe that AnswerFinder, by supplying both exact answers and supporting snippets, is closer to what users would expect of question answering systems then the other web based QA systems we have evaluated – although the actual performance of some of the systems (notably PowerAnswer) far outstrip that of AnswerFinder over the TREC test sets.

In a brief experiment to determine the relative performance of the available online QA systems we put the fifty questions used to evaluate web search engines in Section 7.2 to AnswerBus, AnswerFinder, IONAUT and PowerAnswer[8].

[7] `http://www.ionaut.com:8400`

[8] The questions were presented to the four systems on the same day, within as short a period of time as was possible, so that the underlying document collection, in this case the web, would be relatively static and hence no system would benefit from subtle changes in the content of the collection. Note also that this is a single informant evaluation performed by the author.

Firstly we should make it clear that this comparison is a little unfair, as two of the systems, AnswerFinder and IONAUT, return exact answers with supporting snippets while AnswerBus and PowerAnswer perform the easier task of returning answer bearing sentences. Given this, however, it should be clear from the results in Figure 9.2 that AnswerFinder performs exceptionally well over this small set of questions out-performing all but PowerAnswer which is only providing relevant sentences. On the other hand this suggests that AnswerBus is actually performing quite badly as it is performing poorly in comparison to AnswerFinder (and PowerAnswer) and is also only attempting the easier task of retrieving relevant sentences.

It should be noted that due to the small number of test questions it is difficult to draw firm conclusions from these experiments. The results are encouraging, however, given that the average performance of systems evaluated as part of the QA track at TREC 2002 was only 22%, i.e. only 22% of the questions were correctly answered at rank one, hence both AnswerFinder and PowerAnswer have a performance well above average.

Part III

How To Define A Term

Chapter 10

Introduction to Definitional QA

Certain types of question cannot be answered by a single exact answer. For example questions such as *"What is aspirin?"* and *"Who is Aaron Copland?"* do not have a single short answer. Answers to such questions should resemble entries in encyclopaedias or biographical dictionaries telling the user all the important information about the subject of the question. This type of question is usually referred to as a definition question.

While definition questions can be natural language questions there are only a few ways in which they can phrased and most are of the form *"Who/What is/was X?"*. In these questions *X* is the *definiendum* often referred to as the 'target' of the question and is the thing, be it person, organization, object or event, which the user is asking the system to define.

Embedding the target in a question seems artificial. Users of electronic encyclopaedias would not expect to have to enter a full question to find an article but would usually enter just the name of the thing they were interested in. While more natural to the user it actually complicates the problem for researchers designing definitional QA systems. With full questions it is easier to discern if the target is a person or some other entity allowing a definition for a person to be constructed differently to those for an organization or generic name, such as aspirin. By taking only the target as input there is no obvious sign of what type of thing the target is (i.e. no words like *who*) and as such all targets are likely to be treated the same. The research into answering definition questions presented in the following chapters will assume that the input to the system will be just a target which is not embedded in a question.

As currently the only accepted test sets for definitional question answering are those used in the TREC evaluations (from 2003 onwards) we will assume the same scenario for guiding the production of a definition (Voorhees, 2003b):

> *The questioner is an adult, a native speaker of English, and an "average" reader of US newspapers. In reading an article, the user has come across a term that they would like to find out more about. They may have some basic*

idea of what the term means either from the context of the article (for example, a bandicoot must be a type of animal) or basic background knowledge (Ulysses S. Grant was a US president). They are not experts in the domain of the target, and therefore are not seeking esoteric details (e.g. not a zoologist looking to distinguish the different species in genus Perameles).

As detailed in Chapter 3 the currently accepted evaluation methodology for definitional questions focuses on the inclusion in the definition of given nuggets of information. For example when asked to define *"Bill Bradley"* systems should, according to the TREC supplied answer key, include in a definition the following facts: basketball hero, US senator, presidential candidate, Rhodes scholar, and elected to the NBA Hall of Fame. Notice that this does not include a lot of facts that you would expect in a full definition of a person. There is no mention of when or where he was born; given that he is a basketball hero who did he play for; and which state did he represent as a US senator. While the system and ideas developed in the following chapters will be evaluated using the TREC 2003 questions, answer keys, and evaluation metric it is important to remember that the TREC viewpoint may not accurately represent that of real world users. Whilst it may be preferable to update the answer keys so as to more accurately reflect the information usually contained in encyclopaedia or biographical dictionary entries this would result in evaluations which could not be directly compared with previously published research.

While definitional question answering systems tend to adopt a similar structure to systems designed to answer factoid questions (see Chapter 5) there is often no equivalent to the question analysis component due to the little information contained in the question. Any processing of the question that does take place happens within the document retrieval process in an attempt to maintain a useable search space for the extraction of information nuggets.

This chapter briefly outlines the two components of a definitional QA system by way of examples from the literature. The following two chapters then introduce our approaches to document retrieval and nugget extraction for answering definitional questions.

10.1 Document Retrieval

In many definitional QA systems the document retrieval component is responsible for locating sentences which may contain information pertinent to the definition of a target. Many systems locate relevant sentences by firstly retrieving documents which contain any of the words in the target. This is guaranteed to find all documents which discuss the target but will of course find many which have no relation to the target. For example retrieving documents in this way when defining *Alexander Pope* will find all those documents which mention Alexander Pope as well as those talking about the Roman Catholic Pope and those which mention any other person called Alexander. A number of approaches for selecting only relevant sentences have been proposed.

Xu et al. (2003) remove irrelevant sentences by selecting only those which contain the target verbatim or if the target is a person contains the first and last names separated by no more than three other words. This allows them to select sentences containing *George W. Bush* when defining *George Bush*. The QUALIFIER system (Yang et al., 2003) works in a similar fashion although it performs coreference resolution allowing it to also select sentences which refer to the target without explicitly mentioning it – very useful for famous people where often only the surname is used, i.e. *Bush*.

The DefScriber system by Blair-Goldensohn et al. (2003) takes a totally different approach to selecting relevant sentences. They used the rule-learning tool Ripper (Cohen, 1995) to construct a classifier which can automatically identify relevant definitional sentences with an accuracy of 81%.

10.2 Nugget Extraction

Whatever the approach to finding relevant sentences the next stage in most definitional QA systems is to cluster, rank, and simplify the sentences to present a short concise definition.

A number of systems make use of indicative patterns either to select highly relevant sentences in their entirety or to extract short phrases of information. Gaizauskas et al. (2004) look for part-of-speech based patterns to determine relevant facts about the targets. For instance the pattern `TARGET, WD VBD` is used to extract information about people and matches phrases such as *"Aaron Copland, who composed..."*. Xu et al. (2003) uses similar pattern based approaches as well as considering appositives and copula constructions to select relevant phrases such as *"George Bush, the US President..."*.

A number of systems (Echihabi et al., 2003; Gaizauskas et al., 2004) have mined clue words from online bibliographies and dictionaries to allow them to find and highly rank sentences about the target which contain these words. For example, Echihabi et al. (2003) built a list of 6,640 such words which occur at least five times in biographies and which occur more frequently in biographies then standard text. This list includes terms such as; Nobel, knighted, studied, travelled, Composer, edited, and Physicist.

Sentences, or phrases, selected using these (and other) approaches are then ranked usually based upon the features they contain (i.e. the number of clue words or the precision of indicative patterns they match). As the current evaluation metric (see Section 3.1.3) is based partly on the size of the resulting definition ranked sentences are usually clustered and a single representative example from each cluster is used to build the definition. A common approach to clustering is simply the word overlap between sentences with Xu et al. (2003) using a 70% overlap to determine that two sentences contain the same information and Gaizauskas et al. (2004) using 50% overlap.

Chapter 11

Document Retrieval

This chapter is concerned with developing a new approach to document retrieval that can be used as part of a system for answering definition questions. The main problem faced when performing document retrieval for definition questions is the lack of terms which can be used in an IR query. Unlike factoid questions which can contain a number of relevant terms (for example *"What is the highest mountain in England?"* contains the words *''highest"*, *"mountain"*, and *"England"*) which can be used to select relevant documents definition questions consist of just the target of the definition. So when asked to define *"aspirin"* the input to the system will be *"aspirin"*.

This chapter introduces new ideas for dealing with the lack of query terms when answering definition questions as well as a novel approach for handling complex definition targets in a simple yet principled fashion.

11.1 Creating a Suitable IR Query

The document retrieval component is responsible for finding relevant passages given the target of the definition (where passages can be of any length from a single sentence to a full document depending upon the system used). The major problem facing researchers when dealing with definition questions is that little or no information is provided other than the target itself which makes it difficult to construct rich IR queries that will locate relevant documents. For instance submitting the target *Alexander Pope* as two terms to an IR engine will not only retrieve relevant documents but also documents about the Catholic Pope and other people with the name Alexander (31,487 passages in AQUAINT are retrieved by such a query). As the system is attempting to define a given target it is likely that relevant documents will contain the target verbatim in the text and so one possible approach to improving the retrieval and hence the quality of the resulting passages, is to use the target as a phrase when searching a text collection (i.e. in a relevant document both terms must appear next to each other, ignoring stopwords, and in the same order as in the query). For a simple example such as *Alexander Pope* this approach has the desired effect dramatically reducing the number of documents retrieved from the collection while

retaining those which are actually about the target (only 22 passages in AQUAINT contain the phrase *"Alexander Pope"* compared to the 31,487 which contain either or both words).

While this approach has clear benefits for the retrieval stage when dealing with simple examples, more complex targets, for example those with some form of qualification, usually do not appear verbatim in the texts. For example the target *"Ph in biology"* when treated as a phrase does not appear in any of the documents within the AQUAINT collection. In fact of the 43 multi-word targets from the TREC 2003 question set only 36 appear as a single phrase in the AQUAINT collection. Systems need therefore, to process the target in order to weaken the search criteria from a single phrase to an IR query which will allow a system to retrieve at least one relevant document, although due care should be taken to ensure that the system is not swamped by many irrelevant documents. The rest of this section presents an iterative approach to document retrieval that attempts to successively weaken the IR query from a single phrase in a way that retains as much grouping of the terms as possible to facilitate the retrieval of definition bearing texts. The methods adopted were in part motivated by the definition questions from TREC 2003 which are also used to evaluate the system described in this chapter. While it is clear that this is not an ideal approach, the system was built to be as generic as possible. The success of the approach can be seen by examining the performance of the system over the TREC 2004 questions (not available until after the system had been developed) details of which can be found in Appendix E.

Many targets consist of two parts – a mandatory focus and an optional qualification. This separation of the target can be used to produce a weaker yet functional IR query. The system starts by determining if the target is present in an unambiguous form (only one sense) in WordNet, if so then the system assumes that there is no qualification to be removed. If however the term is not present in WordNet then the system attempts to determine which part of the question is the focus and which is qualification. For example the target *"vagus nerve?"* is present in WordNet and so is not examined for the presence of qualification. On the other hand the target *"the medical condition shingles"* is not in WordNet although the focus *shingles*, without its qualification, is defined by WordNet.

Determining which part of the target is the main focus and what is qualification is initially achieved through examination of any differences in case of the words in the target. For example given the targets *"skier Alberto Tomba"* and *"Ph in biology"* it should be clear that the focus in these two instances is *"Alberto Tomba"* and *"Ph"* with *"skier"* and *"in biology"* as qualification. The system examines the tokens in the target one at a time (first from left to right and then from right to left) to spot a change in orthography assuming that a change to uppercase (or at least title case in which the first letter of the token is a capital letter) signifies the start (or end if working backwards through the target) of a proper noun and that the other tokens (those of lowercase or symbols which the system has already examined) are qualification of the proper noun. Note that the system works by spotting a change to uppercase text not just a change in case as some proper nouns contain lowercase words such as *"Friends of the Earth"* which should not be split into *"Friends"* qualified by *"of the Earth"* but should rather be treated as a single proper noun.

Before using the focus and optional qualification to construct an IR query we attempt to remove one other type of qualification. The system checks to see if the target is of the form *X in Y*. Usually when a target is of the form *X in Y* then *X* is usually the focus, while *Y* is qualification. For example *Abraham* is what the system is being asked to define when given the target *'Abraham in the Old Testament'* with *'in the Old Testament'* as qualification of which specific Abraham should be defined.

At this stage we have a refined focus and a set of qualification words or phrases which the system can use to build an IR query. The IR query is constructed by treating the focus as a phrase and combining all the qualification terms using the boolean `AND` operator. For example *'skier Alberto Tomba'* becomes `skier AND 'Alberto Tomba'` while *'Abraham in the Old Testament'* becomes `Abraham AND 'Old Testament'`[1]. This allows the system to retrieve passages, such as the following for the query `skier AND 'Alberto Tomba'` (taken from document APW20000320.0036), which are relevant but would not have been retrieved and considered part of the definition using the original target:

> *"He's a terrific **skier**, who still can win a lot in the coming years," said three-time Olympic champion **Alberto Tomba**, who won his only World Cup overall title and two discipline titles in the cup finals here in 1995.*

If such a refined query results in at least one document being retrieved from the document collection then these documents are passed onto the nugget extraction component and no further processing of the target is carried out.

If, on the other hand, the IR query has not retrieved any documents as it is still too restrictive then a number of further processing steps are available to further weaken the query in order to allow at least one document to be retrieved from the collection.

The next stage is to determine if the target is of the form *X the Y* as this usually signifies a person with a qualification, for example *"Vlad the Impaler"* and *"Akbar the Great"*. If this is indeed the case then *X* is extracted as the focus and *Y* is added to the set of qualification terms, and a new IR query is generated as before (the focus combined with all the qualification words/phrases using the boolean `AND` operator).

If this still does not result in any documents being retrieved then the query is significantly weakened by not requiring all the qualification words to be present in the retrieved documents. If we look again at the Ph in biology example it turns out that using a query requiring both Ph and Biology, `Ph AND biology` does not result in any documents being retrieved. This stage weakens this query to `+Ph OR biology` – this translates to select all those documents which contain the term Ph and when ranking take into account any instances of biology in the documents. Now in fact we know that no documents contain both Ph and biology but in a more complex example (although none have yet

[1] Note that as stopwords are not used by the indexed document collection the words *in* and *the* are not part of the refined query.

been seen) it could be that the system has a number of qualification terms and by relaxing the constraint in this way we allow documents containing the focus and one or more of the qualification terms to appear before those which contain only the focus, while not requiring any document to contain the focus and all the qualification terms.

If the IR query still does not retrieve any documents from the collection then we start again with the original target and take a simpler approach to generating a query. This simply assumes that the last token in the target is the focus and all the other terms are qualification. For example none of the processing so far results in any documents being retrieved for the target *"the medical condition shingles"*. This simpler approach simply forms the query `medical AND condition AND shingles` which is much less restrictive than insisting that all three words appear as a contiguous sequence of words.

If this much simpler query is still too restrictive then it is again simplified by only requiring the final term target (the focus) to be present. This proved useful not only for dealing with the target *"the medical condition shingles"* but also when defining *"Antonia Coello Novello"* as her full name does not appear in the AQUAINT collection. Weakening the query to `Antonia OR Coello OR +Novello` results in selecting relevant documents containing other forms of her name such as Antonia Novello or just Novello, while ranking those documents containing more elements of her name higher than those which contain only her surname.

If this very simple query results in no documents begin retrieved from the collection then the system responds by telling the user that it was unable to build a definition as it could not correctly determine the focus from the target being defined.

This method correctly constructs IR queries for the 50 definition questions used in the TREC 2003 evaluation, although this is not surprising as these questions were used during development. Using this method to build queries for the sixty-four *other* questions used in the TREC 2004 evaluation results in 63 correctly constructed queries. The one target for which the system was unable to construct an IR query resulting in at least one document being retrieved was Q45.4 *"International Finance Corporation (IFC)"*.

Once relevant documents have been retrieved they are split into separate sentences and only those sentences which contain the focus of the target are passed to the nugget extraction component detailed in the next chapter.

We believe this approach to determining the focus and qualification within a target, for the purpose of retrieving relevant text, to be the first to consider the structure of the target in a principled fashion. Previously reported work has used the target in a number of ways. Greenwood and Saggion (2004) insist that the entire target be present in any retrieved documents – a method we previously highlighted as being too restrictive. Yang et al. (2003) select documents which contain all the bi-grams in the target. While this is a more flexible approach it does not take the structure of the target into account in any way. Closest to the work we have presented is the system developed by Xu et al. (2003). For *who* questions only their system performs limited processing of the

target by selecting those documents which contain the first and last token of the name separated by no more than three words. This allows the system to select documents containing *George Walker Bush* when defining *George Bush*. While more principled than the other approaches this has two shortcomings. Firstly the system needs to know that the target being defined is a person. While this was easy to ascertain for the questions in the TREC 2003 evaluation, which were presented as fully formed questions such as *"Who is Aaron Copland?"*, more recent evaluations have used just a target (i.e. *Aaron Copland*) making it much harder to determine that the definiendum is a person. Secondly, and more importantly, this approach fails for targets such as *Vlad the Impaler* or *Abraham in the Old Testament* where there is no last name but rather a first name and an identifying description.

Our approach can be classified as recall enhancement, however, it differs significantly from standard recall enhancement techniques such as relevance feedback (Salton and Buckley, 1990), local context analysis (Xu and Croft, 2000), or secondary term expansion (Saggion and Gaizauskas, 2004) in that it does not involve query expansion. Rather than expanding queries our approaches relies on query refinement by altering the structure of the query (e.g. term grouping) to improve the IR performance.

11.2 Higher Quality IR via WordNet Glosses

The previous section highlighted the fact that retrieving relevant documents or passages for definition targets, even those which appear verbatim in text, can be challenging due to the lack of rich context provided in the question. This problem is compounded by that fact that just because a document contains the focus of the target it may not actually define any aspect of it.

The approach taken here is to use WordNet as a source of information from which the system can extract terms relevant to the focus of the target so as to provide a richer IR query (for an approach which utilizes trusted web sites see Greenwood and Saggion (2004)). Once the focus of the target has been identified (i.e. the qualification has been removed) then WordNet is consulted for a definition. If the focus of the target appears in WordNet and has only one sense (i.e. it is unambiguous) then the gloss (WordNet's definition of the term) is extracted and all terms in the gloss are added to the IR query to produce a richer IR query. For polysemous words the IR query is not expanded as adding words from multiple definitions may have an adverse affect on the retrieved documents. For instance, WordNet lists 7 different senses for the word battery, adding the glosses of all seven definitions is unlikely to be helpful.

A nice example of this approach is the target *"Ph in biology?"* from which the methods detailed in the previous section correctly determine that *Ph* is the focus while *"in biology'* is qualification. 559 passages are retrieved from the AQUAINT document collection using the query generated for this target. The focus *Ph* has a single unambiguous entry in WordNet:

> *(chemistry) p(otential of) H(ydrogen); the logarithm of the reciprocal of hydrogen-ion concentration in gram atoms per liter; provides a measure on a scale from 0 to 14 of the acidity or alkalinity of a solution (where 7 is neutral and greater than 7 is acidic and less than 7 is basic)*

Adding all of the terms in this WordNet gloss to the IR query does not result in more passages being retrieved from AQUAINT (this method never results in more passages as we still require the target to appear in all retrieved passages) rather it affects the outcome of the ranking algorithm so that longer documents[2] which mention the related terms from the gloss, as well as the focus, are now ranked highly. When using just the focus and its qualification the first passage retrieved from AQUAINT is (NYT19990501.0252):

> *"This is in range with proper pH levels for your pool," he said, explaining that the proper pH range 7.2 to 7.8 can be obtained by adding to the water chemicals known as pH increasers or pH decreasers, which are available at pool stores and home centers.*

This passage although of a reasonable length tells us very little about Ph and what it means, i.e. it is not a very good definition. If we enhance the IR query using the WordNet gloss then the first retrieved passage is much more informative (NYT20000323.0009):

> *The standard gauge of acidity known as pH, for example, ordinarily runs from 0 to 14, where zero is considered the strongest acid, 7 is neutral and 14 is the strongest base.*

In fact this single sentence is an almost complete definition of Ph. Unfortunately it is not sensible to use this approach while retrieving all the passages required by the nugget extraction component. It is unlikely that the WordNet gloss will provide a complete definition of the target (i.e. not all the nuggets identified as valid will be present in the gloss) so using the gloss in this way skews the ranking of documents giving those similar to the gloss first. While this has the affect of making sure relevant documents appear high in the ranking it actually means that other equally relevant passages are moved down the ranks, i.e. using this approach would allow the system to accurately define only one aspect of a target. As a trade off the system retrieves the first 3 passages from AQUAINT using the WordNet gloss and then reverts to using the focus and any known qualification from the target. The nuggets selected when IR is performed using a WordNet gloss are heavily weighted to ensure that they appear at the beginning of the definition[3].

[2] The nugget extraction process in the following chapter relies partly on the order in which the passages are ranked by the IR engine. Since most IR systems employ some form of length normalization function this results in short passages (often just article titles) appearing towards the top of the ranking. These short passages tend to be of little use in defining a target but nevertheless tend to find their way into the definition which is why retrieving longer documents is preferred.

[3] The number of passages to retrieve using the gloss as context and the nugget weighting were determined via manual inspection of a few example questions and may differ from one system to another, especially between those systems retrieving full documents and those performing passage retrieval.

It may seem strange to use a dictionary definition, such as a WordNet gloss, to locate definition bearing sentences – if all the user wants is the definition of a term then why is a dictionary definition unsuitable. There are two reasons why it is useful; evaluation and completeness.

If we wish to compare two definitional QA systems then it is likely that they will be required to draw their answers from the same document collection. For example the TREC evaluation of definition systems insist that each nugget of information is related to a document in the AQUAINT collection. In this case the systems cannot simply return a dictionary definition of the term but they can use the definition to find similar sentences within the collection.

The other reason for using dictionary definitions in this way is completeness. If a system is asked to define a term which could change then it maybe that the dictionary definition is either out-of-date or incomplete. Using the definition as a starting point would therefore allow the system to also find contradicting (a contradicting text is highly likely to contain the same terms as the definition) or more complete texts and hence produce a better more complete definition than the dictionary alone could have provided.

Evaluation of using WordNet to improve the selection of relevant documents when answering definition questions is given in Section 12.3 of the following chapter.

11.3 Limitations of the Approach

The one main issue which has not been addressed in this work is that of relevant information being spread across multiple sentences. As was previously mentioned only those sentences which actually contain the focus of the target are passed to the extraction component and as the system makes no attempt to perform coreference resolution or any discourse interpretation and so any information which can only be found by relating a number of sentences is lost. The extent to which this is a problem is currently unclear although in a recent study Stevenson (2004) showed that in the MUC-6 data set (Sundheim, 1995) only about 40% of the known event relationships were fully contained within a single sentence. If the same applies to definitional nuggets then clearly future work should concentrate on extending the retrieval component to include coreference resolution so as to allow other relevant sentences, containing a pronoun or other relational phrase instead of the focus of the target, to be considered as part of the final definition.

The main reason for not performing more intensive coreference resolution is that while the number of documents from which nuggets can be extracted will be increased so will the time taken to process the documents. Instead of only requiring tokenization and sentence splitting each document would also require POS tagging, named entity recognition and coreference resolution which may mean that the approach would no longer work within a reasonable time frame.

11.4 Summary

This chapter has introduced a straightforward approach to document retrieval for answering definition questions. The approach to analysing the target to determine both the main focus and optional qualification we believe to be novel, more principled than previous approaches, and applicable to a wider range of targets while requiring no prior knowledge of a target's semantic type (i.e. all targets are analysed in the same way be they people, companies or *'things'*).

Using WordNet to expand the limited query terms for definition questions is not a new technique, but by limiting the use of this resource to both unambiguous targets and for the retrieval of only a few documents we avoid constraining the definition to just the aspects of the target mentioned in WordNet.

As the techniques for both determining the focus of the target and expanding the retrieval using WordNet are straightforward and do not require intensive processing to perform they are ideally suited to use in real-time QA systems.

Chapter 12

Nugget Extraction

The approach to extracting nuggets from relevant passages detailed in this chapter was in part inspired by the baseline system submitted by BBN as part of the TREC 2003 evaluation (Xu et al., 2003). Their baseline system builds a definition by retrieving sentences from the document collection which contain the target (by "contained" they mean the sentence includes a noun phrase that matches the target via string comparison or coreference resolution) and adding each such sentence to the definition if it does not overlap more than 70% with any of the sentences already in the definition. Retrieval stops when the resulting definition consists of at least 4000 non-white space characters or there are no more relevant documents to process. This simple system achieved an F(β=5) score of 0.49 and was out performed in the TREC 2003 evaluation only by the other runs submitted by BBN (Xu et al., 2003).

The nugget extraction system presented in this chapter builds upon the ideas behind the (Xu et al., 2003) baseline to produce a system which outperforms it while still requiring very little time to process each question. The aim of the approach is to produce a concise, easily readable definition from the sentences retrieved using the document retrieval strategy outlined in the previous chapter. To this end we borrow a number of ideas from both the summarization community and good writing style guides. Together these allow us to implement a number of methods for simplifying text without altering the meaning or readability.

12.1 Removing Redundant Nuggets

This approach to nugget extraction treats each sentence as a separate nugget of information. The system does not apply deep linguistic processing to determine the semantic similarity of sentences but an approach is required to determine if any two sentences contain *roughly the same* information. That is, given sentence A does sentence B provide any new and relevant information for the purpose of defining a given target.

12.1.1 Word Based Sentence Filtering

Two different approaches for determining if sentence B contains novel information about the target when compared to sentence A were implemented: word overlap and cosine similarity (for a good overview of cosine similarity see Witten et al. (1999b)). Clearly other approaches could also be used to compare sentences including containment or resemblance (Broder, 1997).

Word Overlap As previously stated the original motivation for the approach to definition creation detailed here was the surprising performance of the TREC 2003 baseline system (Xu et al., 2003), which achieved an F(β=5) score of 0.49. The baseline system determined if a new sentence was novel by comparing its percentage word overlap with the set of sentences already added to the definition – if the overlap was greater than 70% then the sentence was discarded as unlikely to contain any novel information.

A sentence profile is constructed for each sentence which contains the set of stemmed (Porter, 1980) non-stopwords, T, in the sentence. The similarity, S, between sentences A and B is then formally defined as:

$$S(A,B) = \frac{|T_A \cap T_B|}{min(|T_A|, |T_B|)} \tag{12.1}$$

This is the percentage of tokens in A which appear in B or the percentage of tokens in B which appear in A, whichever is larger.

A number of small experiments using this measure determined that using a cut-off point of 0.70 (as in the original baseline) results in the best performance.

Cosine Similarity with TF.IDF Weighting The cosine similarity measure is usually used in conjunction with a term weighting function. In this experiment we make use of the well known TF.IDF weighting function. Let $f_{t,d}$ be the frequency of term t within document d while n_t is the number of documents in a collection of N documents containing term t. The TF.IDF weight $W_{t,d}$ of t within d is then defined as:

$$W_{t,d} = log(f_{t,d} + 1) log\left(\frac{N}{n_t}\right) \tag{12.2}$$

Given this weighting function the cosine similarity S of sentences A and B is given by Equation 12.3:

$$S(A,B) = \frac{\sum_{t \in A \cap B} W_{t,A} W_{t,B}}{\sqrt{\sum_{t \in A} (W_{t,A})^2} \sqrt{\sum_{t \in B} (W_{t,B})^2}} \tag{12.3}$$

In these experiments sentences A and B were considered similar if the cosine similarity was above 0.2 (as suggested by Salton et al. (1997)). The cosine similarity is relatively easy to compute over a collection for which we have access to an inverted index, but we have to approximate values for both N and n_t if working with collections such as the web for which these figures are not easily obtainable.

Max Length	Word Overlap	Cosine Similarity	
1000	0.4192	0.3716	▽
2000	0.4477	0.4230	
3000	0.5130	0.4431	▼
4000	0.5176	0.4503	▽
5000	0.5165	0.4596	▽

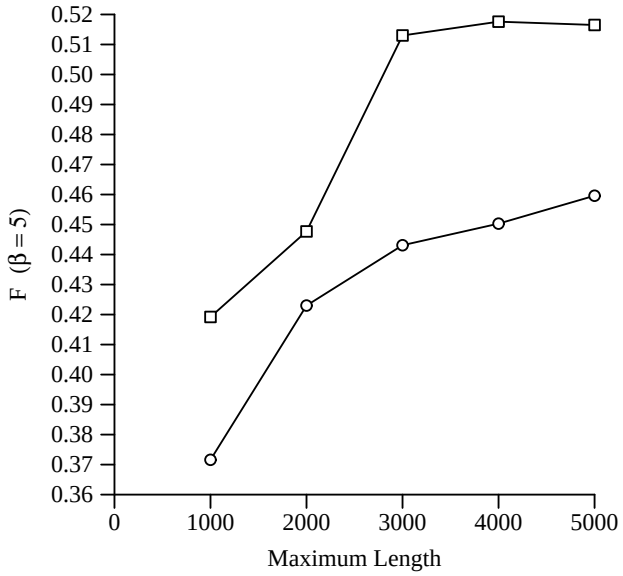

Figure 12.1: Comparison of step 1 filtering approaches, word overlap, □, and cosine similarity, ○.

The performance of these two approaches to similarity detection over the fifty TREC 2003 definition questions can be seen in Figure 12.1 (see Chapter 4 for full details of this experimental setup which is used throughout this chapter). These results show that the simple word overlap is better than the cosine similarity measure for determining if a new sentence contains unseen information (this could in fact be due to the use of the TF.IDF weighting function rather than the cosine measure in general). Using word overlap in this way results in a system which is very similar (although with a more principled approach to determining the focus of the target as detailed in the previous chapter) to the baseline system submitted by BBN to the TREC 2003 evaluation. That baseline achieved an F(β=5) score of 0.49 which is similar to the score of 0.5176 achieved here when a definition of up to 4000 non-whitespace characters is constructed. The rest of this chapter will now build upon the word overlap approach through further sentence filtering and redundant phrase removal techniques.

12.1.2 n-gram Sentence Filtering

Both methods of determining similarity suffer from the same problem – the notion of similarity we are using is very crude. Often two sentences will tell the user the same piece of information about the target while also containing a lot of additional irrelevant information. The ideal similarity test would be resilient to irrelevant information in the sentences, but such a similarity test would require deep linguistic understanding of the sentences to determine their meaning and hence the information specific to the target.

The problems involved in determining if a new sentence tells the user anything that is both new and relevant about a target can be illustrated through two sentences relating to the target *"Who is Alice Rivlin?"*:

Federal Reserve Vice Chairman Alice Rivlin has said in more than one inter-

view recently that the current stock market is high by any kind of valuation.

Last week, Alice Rivlin, whom Clinton appointed vice chairman of the Federal Reserve, announced her resignation from the central bank effective July 16.

While it is true that these two sentences are different given any standard measure of similarity, both contain only one identical piece of information pertinent to the task of defining Alice Rivlin, namely that she is (or was) the vice chairman of the Federal Reserve. The other information in the sentences, while not important to the definition of *Alice Rivlin*, causes the two sentences to be considered different by both of the similarity measures previously mentioned (word overlap and the cosine measure) leading to redundancy in the generated definition.

The two example sentences share only the four words *federal*, *reserve*, *vice* and *chairman* and therefore the word overlap, using Equation 12.1, is 36.4% (after having removed stopwords and the target, Alice Rivlin, as this is common to all sentences and so should not be used for similarity determination). Such a low overlap certainly does not immediately suggest that the two sentences are similar enough for one or the other to be discarded. One assumption we can make is that similar sentences will contain similar words in a similar order. So one possible way of determining similarity would be to look for n-grams in the sentences, i.e. sequences of multiple words appearing in both sentences – this is the same assumption underlying the Rouge automatic scoring system detailed in Section 3.1.3. In the two example sentences we can see that they both contain the bi-grams *'federal reserve'* and *'vice chairman'*, giving a bi-gram overlap of 20% (the shorter of the two sentences contains 10 bi-grams). We can use this notion of overlapping n-grams to implement a measure of similarity designed to filter out the remaining similar sentences not handled correctly by the word overlap measure.

The filter works by calculating the sum of the percentage overlap of increasingly longer n-grams. The n-grams considered range from length one (a single word) to length s which is the length, in words, of the shortest of the two sentences being compared, with $N_{n,A}$ being the set of n-grams in sentence A. The similarity measure is formally defined as:

$$sim(A, B) = \sum_{n=1}^{s} \frac{|N_{n,A} \cap N_{n,B}|}{min(|N_{n,A}|, |N_{n,B}|)} \tag{12.4}$$

From limited testing, a cutoff level of 0.5 was determined with pairs of sentences having a score above this deemed equivalent. Given that there are no common tri-grams between the two example sentences the similarity score calculated using Equation 12.4 is 0.564. This similarity score is above the threshold and so the two sentences are deemed equivalent and so one of them can be discarded.

Figure 12.2 shows the results of applying this second filter to the sentences filtered using word overlap. Filtering in this way results in a consistent (sometimes statistically significant) improvement over the standard word overlap approach.

Max Length	Word Overlap	n-gram Overlap
1000	0.4192	0.4276 △
2000	0.4477	0.4660 △
3000	0.5130	0.5162
4000	0.5176	0.5199
5000	0.5165	0.5259 ▲

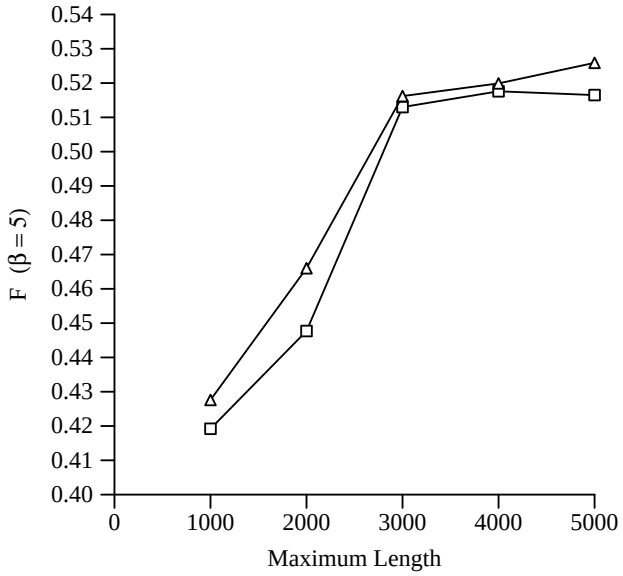

Figure 12.2: Comparison of n-gram overlap filtering, △, with 70% word overlap, □.

For an example of the usefulness of this second filter we can look again at the content of the sentences returned by the system when asked to define *Alice Rivlin*. Using just word overlap to filter the sentences leaves eighteen sentences (within the 5000 length cut-off) which state that Alice Rivlin is or was the Federal Reserve Vice Chairman. Filtering the sentences remaining after word overlap has been used to remove redundant information using the n-gram overlap method reduces the number of sentences stating that Alice Rivlin is the Federal Reserve Vice Chairman to just seven without removing any other relevant nuggets from the final definition.

12.2 Removing Redundant Phrases

Within any passage of text it is quite likely that some of the words or phrases will be redundant; that is, removing them has no effect on the meaning of the text. As has already been stated, answers to definition systems should be concise passages of text and therefore any extraneous words or phrases, which can be removed, should be removed. This will have the effect of increasing the amount of relevant information within a given length of text (a side affect of this should be an increase in the F-measure score given the current evaluation metric, see Section 3.1.3).

12.2.1 Locating Redundant Phrases

As we wish for the approach to be capable of generating definitions quickly we attempt to find redundant words or phrases which can be identified using predominately shallow methods. This rules out approaches to detection of redundant phrases which would require full syntactic or semantic parsing to identify. A number of sources were consulted for possible ways in which redundant phrases could be both identified and safely removed (Dunlavy et al., 2003; Purdue University Online Writing Lab, 2004) which led to a num-

ber of approaches aimed at reducing the amount of text without changing the overall meaning of the sentences or their readability[1]. The approaches to redundancy removal, which we investigated, fall into two categories: those which delete words or phrases and those which transform phrases. Seven different approaches involve the deletion of text, these are:

Imperative Sentences: If the sentence is an imperative, i.e. the first word in the sentence is a verb, then it is discarded. Such sentences are discarded because they are only relevant and informative if they reference information the reader already knows, in which case they are unlikely to add anything new to the definition.

Gerund Clauses: These often comment on rather than add to the content of a sentence and therefore tend not to include information essential to a definition of the target. To identify and remove a gerund clause, it must 1) be at the start of the sentence or immediately following a comma, and 2) have the gerund as the lead word, or as the second word following a preposition, 'while' or 'during'. The end of the clause is identified by a comma or period. The following examples (modified from the example given in Dunlavy et al. (2003)) illustrate all three forms of gerund clause that we can identify and hence remove:

> *While carrying passengers from the Estonian capital Tallinn to Stockholm,* more than 800 lives were lost when the ferry sank within minutes early yesterday morning in the Baltic Sea 40 km south west of the Finnish island of Uto.
>
> More than 800 lives were lost when the ferry, *carrying passengers from the Estonian capital Tallinn to Stockholm,* sank within minutes early yesterday morning in the Baltic Sea 40 km south west of the Finnish island of Uto.
>
> More than 800 lives were lost when the ferry sank within minutes early yesterday morning in the Baltic Sea 40 km south west of the Finnish island of Uto*, while carrying passengers from the Estonian capital Tallinn to Stockholm.*

Gerund clauses are only deemed redundant if they consist of less than half the number of words in the sentence and do not themselves contain the focus of the definition. Longer gerund clauses are retained either because the clause is important or more likely we have incorrectly identified a non-gerund clause which should not be removed.

Leading Adverbs: In some sentences the lead word does not actually contribute to the meaning of the sentence. Certainly, given a sentence in isolation, the words 'and' and 'but' at the beginning of the sentence can be safely removed. Similarly, adverbs when

[1] In the examples which follow redundant text is written in *italics*.

they appear as the first word in a sentence can be safely removed. More importantly if a definition is constructed from a set of independent sentences the presence of these words at the beginning of sentences can often disrupt the flow of information (this is more of a problem in summarization work in which the summary is evaluated in its entirety, unlike definitions in which each sentence is evaluated independently).

Sentence Initial Expletives: These are phrases of the form *it + be-verb* or *there + be-verb*. Such phrases can be useful in expressing emphasis but usually result in longer sentences than is strictly necessary to convey the information. For example the sentence "*It is* the governor *who* signs or vetoes bills." can easily be re-written, without changing the meaning of the sentence, as "The governor signs or vetoes bills.". The system identifies the most common form in which a sentence initial expletive is followed by a noun and a relative clause beginning with 'that', 'which', or 'who'. The sentence is shortened by removing both the expletive and the relative pronoun making the noun the subject of the sentence.

Further examples include, "*There are* four rules *that* should be observed." which becomes "Four rules should be observed." and "*There was* a big explosion, *which* shook the windows, and people ran into the street." which becomes "A big explosion shook the windows, and people ran into the street.". From these examples it is clear that no information has been lost and the results are shorter sentences which are still well formed and easy to read.

Redundant Category Labels: Certain words imply their general categories and so a sentence does not usually have to contain both the word and it's category label. For example, users will know that pink is a colour and that shiny is an appearance. This allows us to shorten sentences by dropping the general category term leaving just the specific descriptive word. So we can re-write the sentence "During that time period, many car buyers preferred cars that were pink *in colour* and shiny *in appearance*." as "During that period, many car buyers preferred cars that were pink and shiny." without altering the meaning of the sentence.

We locate redundant category words by investigating phrases of the form *X in Y* or *a X Y* to locate those in which *Y* is the category of *X* (i.e. "pink in colour" or "a pink coloured"). The system determines if *Y* is the category of *X* by seeing if there is either a hypernym or attribute relationship in WordNet between *X* and *Y*. For example colour is a hypernym of pink and so the system can determine that colour is in fact a redundant category label and remove it.

Unnecessary Determiners and Modifiers: Sentences sometimes contain one or more extra words or phrases which seem to determine narrowly or modify the meaning of a noun without actually adding to the meaning of the sentence as a whole. Although these words or phrases can, in the appropriate context, be meaningful they can often be eliminated. For example "Any *particular type of* dessert is fine with me." can easily be re-written as

"Any dessert is fine with me." without any alteration in the meaning of the sentence.

The system contains a list of 12 such redundant words or phrases (Purdue University Online Writing Lab, 2004) which are simply removed when ever they occur in sentences considered during definition creation.

Unnecessary That and Which Clauses: When a clause is used to convey meaning which could be presented in a phrase or even a single word the sentence length is increased without any increase in the information conveyed to the reader. Often the unnecessary clauses are of the form *that + be-verb* or *which + be-verb* which can be easily simplified. For example "All applicants *that are* interested in the job must..." can be simplified to "All applicants interested in the job must..." without any change in the meaning of the sentence. The system converts unnecessary modifying clauses into phrases or single words by removing *that + be-verb* or *which + be-verb* from before nouns or verbs.

There are two further approaches to redundancy removal which involve the transformation rather than just deletion of phrases, these are:

Circumlocutions: These are indirect or roundabout expressions of several words which can be written more succinctly, often as a single word. It is usually possible to replace both *"the reason for"* and *"due to the"* simply with the word *because*. Unfortunately there are no hard and fast rules which state exactly which expressions can be replaced or with what. For instance, the previous two examples could, in most instances, also be replaced by the word *since*.

The system currently replaces a fixed set of phrases with the words *about*, *when* and *can* (currently only 11 such phrases are replaced (Purdue University Online Writing Lab, 2004) due to the difficulty in determining the correct replacement word).

Noun Forms of Verbs: Sentences which use the noun form of a verb often contain extra words (often the verb *be*) to allow the text to flow correctly. Changing these nouns back to their verb forms therefore reduces the length of the phrase. For example "The function of this department is *the* collect*ion of* accounts." can be reduced to "The function of this department is to collect accounts.".

The system replaces the noun forms with their verb equivalents by looking for phrases of the form *is the X of* and if the verb form of *X* is known then this is replaced by *is to X-verb*. The mapping of nouns to their verb forms is carried out using a mapping pre-compiled[2] from the morphological information in the CELEX database[3] (Burnage, 1990).

[2] Thanks to Wim Peters for providing this mapping.

[3] Linguistic Data Consortium (`http://www.ldc.upenn.edu`) catalogue number LDC96L14.

12.2.2 Performance Effects of Redundant Phrase Removal

Removing redundant phrases would seem to be a sensible thing to do as it allows the inclusion of extra, possibly relevant, information within the same volume of text. However, we need to know both how effective redundant phrase removal is and at what point in the process of building a definition it should be applied.

Phrase removal could be performed as either a pre- or post- processing task. Pre-processing involves removing the redundant phrases from each sentence as they are retrieved from the document collection and before any similarity tests are performed. Post-processing to remove redundant phrases, however, is applied to the final definition, i.e. only to those sentences deemed to be novel in the context of the definition. While post-processing is more efficient as less text has to be processed (only those sentences which made it past the similarity tests rather than all sentences retrieved from the collection) it may result in the removal of text that was used to determine that sentences in the definition contain novel information.

Initial work with removing redundant phrases was carried out as a post-processing step. The motivation for this was due in part to the work of Dunlavy et al. (2003) upon which some of the phrase removal development was based but also as it is easier to view the effect of individual strategies when used as a post-processing task. Removing redundant phrases before filtering the sentences can have an effect on the similarity of two sentences and so can change which sentences are included in the final definition making it difficult to judge the performance of a specific approach to phrase removal.

Common sense would suggest, however, that removing redundant phrases from sentences should be carried out as a pre-processing step so that the selection of relevant sentences is based upon the text that will eventually appear in the definition and not on text that may be removed by later processing.

Figure 12.3 shows the performance of the system with no phrase removal and with the phrase removal as both a pre- and post-processing component. Note that the imperative sentence elimination rule was applied to all three configurations.

These results show that performing phrase elimination as either a pre- or post-processing step improves the performance of the system. The results also show that integrating phrase removal as a pre-processing component within the system is more beneficial than post-processing the text just before delivering the final definition. This makes sense as the selection of sentences to use is then based upon the text the user will see and not on phrases which may latter be removed. A similar trend towards phrase removal as a pre-processing stage (rather than post-processing) has also been observed in the summarization community (Conroy et al., 2004).

It is difficult to determine the exact effect of each proposed method of redundancy removal on the overall performance of the system due to the way in which multiple approaches can be applied to a single sentence and the changes this can make in the similarity of the

Max Length	Redundant Phrase Removal		
	None	Pre	Post
1000	0.4276	0.4219	0.4276
2000	0.4660	0.4651	0.4637
3000	0.5162	0.5237	0.5233
4000	0.5199	0.5398	0.5337
5000	0.5259	0.5397	0.5343

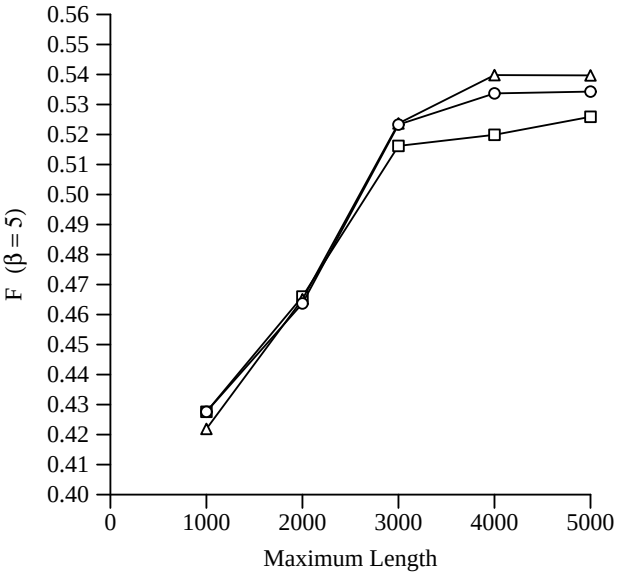

Figure 12.3: Comparison of using pre- (△), and post-processing (○) of sentences to remove redundant phrases with no phrase removal (□).

Phrase Type	Number of Occurrences	% Correct	Average Character Saving
Gerund Clause	125	84.8%	49.32
Leading Adverbs	161	92.5%	4.27
Sentence Initial Expletives	6	16.7%	6.17
Redundant Category Labels	6	100%	6.67
Unnecessary Determiners and Modifiers	50	86.0%	7.94
Circumlocutions	1	100%	10.0
Unnecessary That and Which Clauses	20	95.0%	8.35
Noun Forms of Verbs	1	0.0%	6.0

Table 12.1: Indirect evaluation of phrase removal techniques.

simplified sentences. Two things which it is possible to measure, over and above the end-to-end performance, is the number of times each redundancy approach was applied and what percentage of these removals resulted in badly formed sentences. By badly formed sentences we mean that the sentences are no longer syntactically valid and not that important information relating to the target has been removed. Table 12.1 lists the different approaches and their perceived performance (the removal of imperative sentences is not listed as this was applied to all three configurations given in Figure 12.3) when using the post-processing setup as in Figure 12.3.

It is clear that some of the methods of redundancy removal that were employed perform better than others. Of all the changes made to the sentences not a single instance of important information being removed was found. As the evaluation metric does not specifically require well formed sentences or phrases as answer nuggets (although this was the original aim laid out at the beginning of this section) then the removal of the redundant text is clearly beneficial.

Max Length	WordNet Without	WordNet With
1000	0.3489	0.4219 ▲
2000	0.4305	0.4651
3000	0.5042	0.5237
4000	0.5153	0.5398 △
5000	0.5101	0.5397 △

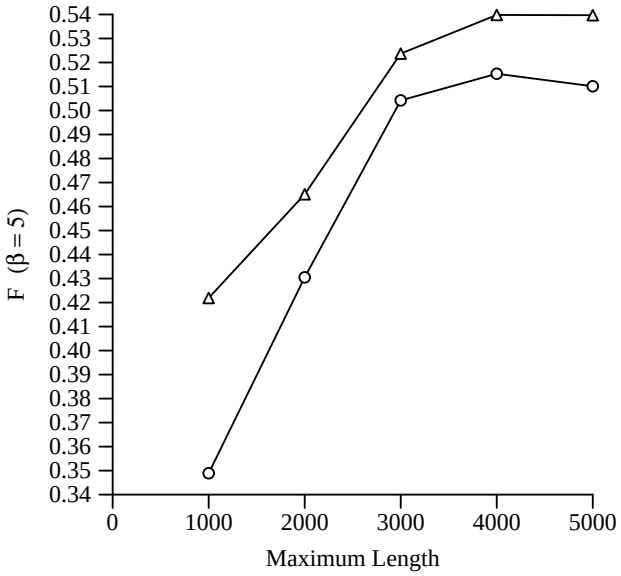

Figure 12.4: With (△) and without (○) using WordNet to improve IR performance.

12.3 The Benefits of IR Using WordNet Glosses

All the results presented in this chapter have made use of WordNet gloss entries, as described in Section 11.2, to ensure that highly relevant documents are examined first when a definition is constructed. As yet, however, no evidence has been presented to show that this approach actually has a beneficial effect on the performance of the system.

Figure 12.4 reproduces the performance of the system when employing redundant phrase removal as a pre-processing task (the best performing configuration discussed in this chapter) along with the results of a system identical in every respect apart from the fact that the IR component does not make use of WordNet glosses (when available) to increase the quality of retrieved documents.

From these results it is obvious that using WordNet in this way has a positive effect on the performance of the system. Unfortunately many questions will not have an entry in WordNet and so will not benefit in the same way although other knowledge sources could be incorporated in a similar fashion to cover a wider variety of questions.

12.4 Producing a Final Definition

The one problem with a simple approach based on retrieving non-redundant sentences is that the resulting definition tends to consist of a large number of sentences (especially if the target is common within the collection). Clearly the ideal system would return a concise paragraph containing all the relevant information about the target and nothing else. That is it would have high recall (mention most if not all of the relevant information) while also having a high precision (contains very little irrelevant information). The evaluation metric used for definition systems (see Section 3.1.3) does indeed take this view although it is currently biased towards preferring high recall over high precision.

	Cut-off Length				
	1000	**2000**	**3000**	**4000**	**5000**
β = **1**	0.3305	0.2914	0.2834	0.2692	0.2639
β = **2**	0.3748	0.3674	0.3816	0.3740	0.3687
β = **3**	0.4003	0.4155	0.4504	0.4527	0.4492
β = **4**	0.4140	0.4430	0.4923	0.5029	0.5013
β = **5**	0.4219	0.4651	0.5237	0.5398	0.5397

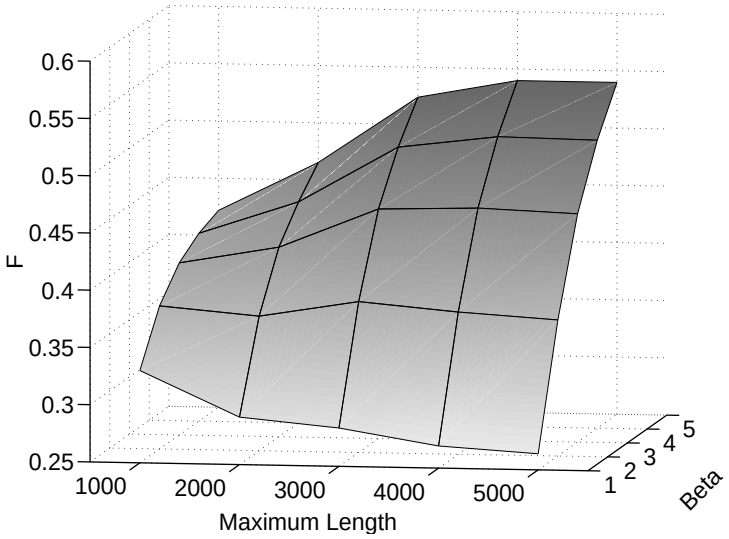

Figure 12.5: Definition F-measure scores for varying cutoff lengths and β values.

Experiments into returning definitions using different cut-off values on the final output length and at different values of β used in the F measure calculation (setting β to 1 means that we give equal weighting to the precision and recall components of the score whereas a β of 5 gives a higher weighting to the recall component) were conducted, using the system described in the previous sections (i.e. the system which produced the with WordNet results in Figure 12.4), to determine the best configuration. The results of different cut-off lengths with different β values can be seen in Figure 12.5.

As can be seen from these results increasing the value of β used in the evaluation metric increases the score independent of the cut-off value examined. This is understandable when one considers that each question has on average eight nuggets accepted as answers which means that on average the evaluation metric gives a maximum precision score if the definition is 800 non-white space characters or less. Clearly all our cut-offs are above this average maximum length and so the system performs better as the bias shifts further towards the recall component of the evaluation and away from the precision component.

12.5 Summary

In this chapter we have introduced techniques from the summarization community and good writing style guides which have allowed us to easily simplify sentences to produce shorter richer definitions of complex targets. The use of predominately shallow techniques

was motivated by the desire to not only develop an effective approach to answering definition questions but to investigate techniques which can be applied within a real-time definitional QA system.

In summary, the approaches presented in this and the previous chapter describe a system capable of quickly answering definition questions. Extending the ideas from the baseline approach (Xu et al., 2003) through the use of WordNet glosses and better methods of sentence and phrase removal has resulted in a system which achieves an F(β=5) score of 0.5348, outperforming all but one system entered in TREC 2003 when evaluated over the same question set. The highest F(β=5) score reported at TREC 2003 was 0.555, achieved by the main system submitted by Xu et al. (2003). Their baseline system, which provided the inspiration for our approach, achieved an F(β=5) score of 0.49. Even evaluating the resulting definitions with a β value of 3 (making the precision component more important) gives a F(β=3) score of 0.4527 which again is only outperformed by one TREC 2003 system when evaluated using F(β=3) (Voorhees, 2003b).

Part IV

Where Do We Go From Here?

Chapter 13

Conclusions

The main motivation behind the work in this thesis was to consider, where possible, simple approaches to question answering which can be both easily understood and would operate quickly – at least in relation to more complex approaches, such as those used in QA-LaSIE (Greenwood et al., 2002). To this end we have introduced a number of novel approaches to answering both factoid and definition questions. It is hoped that these new techniques will advance the field of question answering, giving rise to QA systems which can be used by the general public to access the growing source of knowledge available as free text.

13.1 Factoid Question Answering

Part II of this thesis described QA systems using a three component architecture of question analysis, document retrieval, and answer extraction and introduces a number of approaches to solving the problems associated with each component.

Chapter 6 introduced two approaches to question analysis aimed at determining the expected answer type: one manually constructed and one automatically acquired. The first approach involved manually creating classification rules using a formalism specifically designed for question classification. This approach centred around an answer type hierarchy constructed to cover the semantic entities which we could recognise in free text (rather than all the semantic types we may want to return as answers). The motivation for this is simple – if the system cannot locate instances of a specific semantic type then it will never be able to return answers of that type. Therefore it makes more sense to consider only those semantic types we know how to locate in free text.

Acquiring a question classifier automatically requires a different approach. Whereas the manual approach can be used to build a rule given a single example question, automatic methods require a large amount of hand-labelled training data. Rather than constructing a training and test set to match the answer type hierarchy created for the manual classification approach, we chose to adopt the hierarchy and associated data originally described

by Li and Roth (2002). Using this data we introduced a k-Nearest Neighbours style algorithm for question classification which uses an IR engine to determine similarity between training instances and new unseen questions. This classifier performs exceptionally well when compared with other results reported over the same training and test data. One of the main advantages of this approach is that classifying a new, unseen question occurs in less than 5 milliseconds, and as such is ideally suited to real-time use.

Chapter 7 evaluated a number of approaches to retrieving answer bearing documents from which more detailed processing could later extract exact answers. We showed that while increasing the amount of text retrieved increases the coverage of the retrieved text (the percentage of questions for which at least one relevant document is retrieved, see Section 3.2) it often does so at the expense of the accuracy of downstream answer extraction components. That is, after a certain point the more text an answer extraction component has to process to find an answer, the worse the accuracy. This means that novel approaches to document retrieval have to increase the coverage of a small fixed size volume of text before they are likely to show any increase in end-to-end performance when used within a QA system.

Various methods which attempt to improve the coverage of the retrieved documents were introduced and evaluated. Most of these approaches involved the use of the `alt` operator introduced in Chapter 6 to improve the ranking of documents when expanding query terms. This operator behaves like `or` for the purposes of retrieval but alters the way in which synonymous query terms are scored to better reflect the needs of QA. All the approaches that were evaluated using both `or` and `alt` to group synonymous terms showed that the `alt` operator resulted in a more marked improvement in coverage than could be achieved using the `or` operator. We investigated two main approaches to query expansion: pertainym expansion of location nouns and query expansion via expected answer type. Although both approaches show improvements in coverage of the retrieved documents (in some cases significant), none could be shown to have a significant effect on the performance of an answer extraction component. This is discouraging but it is important to remember that we have assumed a framework of three disconnected components. This separation means that improvements in document retrieval may negatively impact the performance of an answer extraction component which does not take into account the way in which the documents were retrieved. For example, the pertainym expansion of location nouns means that questions such as *"What is the capital of Syria?"* may result in documents being retrieved which talk about the *Syrian capital*. Any answer extraction component which does not take the expansion into account, will not include *Syrian* in any comparison between the question and answer text which could result in the answer being ignored or lowly ranked.

While many researchers have experimented with QA over the web, using techniques developed for closed collections (such as the systems evaluated as part of TREC), to date little consideration has been given to the IR component of such web based systems. Usually a standard web search engine is used as the IR engine in such systems and often just the snippets returned by the search engine are used as relevant passages. In Section 7.2 we evaluated the snippets produced by a number of web search engines to determine

if they were suitable to be used as relevant passages by a QA system. This evaluation showed that examining just the top ten snippets from Google or AltaVista is equivalent in both coverage and answer redundancy to examining the top thirty passages retrieved from AQUAINT. This suggests that web search engines are indeed suitable for use as an IR engine in a QA system.

Chapter 8 introduced two answer extraction components based around semantic type extraction and surface matching text patterns. The semantic type answer extraction component attempts to answer those questions assigned an expected answer type by the manually constructed classifier of Chapter 6. This extraction technique involves extracting from candidate answer bearing documents all entities of the expected answer type and then using a ranking formula and similarity measure to determine which of the extracted entities is the correct answer. As the answer type hierarchy upon which the approach is based cannot be expected to cover all possible answer types, we introduced the idea of expanding the hierarchy using WordNet to increase the number of questions which could be answered. Both of these techniques rely on only shallow processing to extract, compare and rank candidate answers; making them ideal for use in a real-time QA system. The approach was independently evaluated as having an accuracy of 21.3% (see Appendix E) which while not comparable to the current state-of-the-art is promising for such shallow techniques.

The surface matching text pattern approach to answer extraction detailed in Section 8.2 learns rules which encode the local context, linking the main question term to the answer. The context is simply the words appearing between and around the two terms, some of which are generalized to representative tags using a named entity recogniser. Using the surface structure of the text ensures that the patterns can be matched quickly against new texts without any time-consuming processing. For the evaluations reported in this thesis, pattern sets were constructed to cover six different question types. The main drawback to this approach, and the reason that only six pattern sets were constructed, is that they are acquired using a supervised machine learning algorithm requiring hand crafted training data which is expensive to produce. Another limiting factor is that the performance of the pattern sets can vary quite considerably if they are used over writing styles different to those from which they were acquired. In this work, the patterns were acquired from unrestricted text retrieved from the web and when evaluated over other web documents have an average accuracy of approximately 50%. However, when used to answer questions over newswire articles from the AQUAINT collection, the accuracy drops to only 6%. This seems to be because the newswire articles are written in a different style to that of the majority of web documents. This highlights just one problem encountered when attempting to move to QA over the web – techniques which work for one collection of documents may not transfer to another set of documents. This does not mean that we should ignore the technique just because it does not transfer easily to a new collection. Rather, we should remember that evaluations such as TREC, while essential in fostering new research in QA, are not totally representative of what QA should strive to be.

Throughout this thesis we have concentrated on relatively simple approaches to all aspects of QA partly so that systems built using these techniques would be able to answer

questions quickly and could form the basis of real-world QA applications. Chapter 9 combines many of the techniques developed in Part II of this thesis to produce a QA system called AnswerFinder. AnswerFinder answers questions by drawing answers from the web, and in brief testing performs well when compared with other publicly available web based QA systems. The use of shallow techniques results in a system which is capable of answering questions in approximately 8 seconds. While this is still a long time compared with modern web search engines, no attempt was made to optimise the underlying code for speed. What AnswerFinder does show is that the simple techniques for QA introduced in this thesis can be successfully combined to produce a real world QA application.

13.2 Definitional Question Answering

Part III of this thesis described an approach for handling complex definition targets and for finding and simplifying relevant sentences which when combined can be used to produce a definition.

A definition question is not a fully formed question, like a factoid question, rather it is just the name of the thing the user wishes to define. For example *"aspirin"* and *"skier Alberto Tomba"* are both definition questions. Just like factoid questions the target of the definition has to be analysed to enable relevant documents to be located. With no guiding words, such as *who*, *where*, *when*, etc., in the targets to guide the analysis of the targets this can be challenging. Chapter 11 introduced a principled approach to target processing to allow us to find relevant documents. This approach assumes that a target consists of a mandatory focus and optional qualification. We use an iterative approach to determining the focus and qualification at each iteration, seeing if the new division will locate relevant documents. Basing the processing on the way in which targets can be expressed is a more principled approach than using just the words in the target, which we showed leads to better more relevant documents being retrieved. This approach does not rely on any intensive processing of the target and hence it operates quickly.

Our approach to definition creation was inspired by the baseline system entered in the TREC 2003 evaluation by Xu et al. (2003). Having found relevant sentences (using our principled target processing outlined above) we introduced, in Chapter 12, a two stage comparison and filtering approach to select definition bearing sentences. Both these filters are based on overlap between sentences and so do not require deep linguistic processing to extract meaning from the sentences. The first filter assumes that two sentences contain the same information if they overlap, at the word level, by more than 70%. Although this approach can quickly remove very similar sentences, it is incapable of removing sentences which while talking about different subjects contain the same information about the target of the definition. For example, if two sentences mention the job title of the person being defined but are otherwise concerned with different topics it is unlikely that they will overlap more than 70%. The second filter measures n-gram overlap rather than word overlap allowing it to locate sections common to two sentences, e.g. multi-word job titles. Section 12.1.2 gives an example in which this second filter reduces the number of duplicated sentences for an information nugget by approximately 60%.

While these approaches to selecting relevant sentences allow us to construct a definition, the sentences still contain redundant information that it would be useful to remove. To simplify the sentences and hence condense the definition Section 12.2 introduced a number of shallow approaches to locating and removing redundant phrases. Approaches handling eight different redundant phrase types were developed and shown to be effective when used to simplify sentences before filtering.

The approach to answering definition questions detailed in this thesis performed well in independent evaluation as part of TREC 2004, ranking 11th out of the 63 systems which were evaluated (see Appendix E). This shows that not only are the techniques we have developed able to quickly produce a definition (the average time to produce a definition is 21 seconds) but that the definitions produced compare favourably with some of the best definition systems produced to date.

13.3 Challenges Facing Our Approaches

While it is true that the techniques introduced in this thesis have performed well (above average in independent evaluations) it is clear that there are many questions which they are unable to answer correctly. Improvements to the underlying tools, such as the semantic entity recogniser, while probably resulting in slight improvement to the performance would not dramatically increase the percentage of questions which could be answered correctly.

Some of the challenges and limitations of the approaches we have introduced in this thesis have already been discussed along with the techniques themselves. This section will briefly recap the challenges with a view to generating discussion on ways in which the techniques we have introduced can be extended and improved upon.

Chapter 5 introduced a three component architecture for answering factoid questions. This framework divides a QA system into question analysis, document retrieval, and answer extraction. Chapters 6, 7, and 8 introduced techniques for each of these three components. One of the problems with such a framework for QA is the relative independence of the different components. In Chapter 7 we repeatedly stated that while a number of approaches improved the coverage of the documents retrieved for a question they did not result in obvious improvements in the end-to-end performance of a QA system. One of the reasons given for this was that an answer extraction component which is not aware of the IR approach being used will not be able to take full advantage of the improvement in the documents. This suggests that splitting a QA system into separate components may actually decrease the overall performance. While we believe that it is useful to think of a QA system as a pipeline of question analysis, document retrieval, and answer extraction it is clear that in a real-world implementation these components are likely to be highly interlinked. Changes to the way one component operates necessitates altering the succeeding components in the pipeline and this will be addressed in our approaches in the future.

Many of the techniques introduced in this thesis have relied on semantic entity detec-

tion. The manual question classifier of Section 6.1.1 uses a semantic entity recogniser for classifying questions and assigns a semantic type to each question as the expected answer type. The document filtering of Section 7.3 relies on determining which documents contain instances of a specific semantic type. The semantic type extraction component of Section 8.1 can only return things located by the semantic entity detector. This reliance on semantic entity detection, while appearing to produce an effective QA system does limit these approaches. Any entity not correctly detected can never be used by any system. More importantly it would be both difficult and time consuming to produce a recogniser for every possible entity type. This means that the answer type hierarchy behind these approaches is unlikely to be complete and so there may be questions which go unanswered. This does not mean that such approaches should be abandoned but rather other approaches to QA should be developed to supplement the approaches introduced in this thesis.

To reduce the time taken to answer unseen questions we have only considered relatively simple techniques for QA. This has meant that we have applied only tokenization, sentence splitting, and semantic entity recognisers to the texts from which answers are being extracted. This reliance on basic techniques means that information is not considered within a wider context. For example only a naïve attempt was made to handle coreference when answering factoid questions. The approach was simply to consider the preceding sentence as well as the answer bearing sentence if doing so increased the percentage overlap between the context of the candidate answer and the question. Whilst this improves the ranking of the extracted answers it does not allow other candidate answers to be found. This problem was discussed in Chapter 11 as an issue when retrieving sentences for use in definition creation but it also applies to answering factoid questions.

13.4 Future Directions

Whilst the approaches to QA introduced in this thesis allow many questions to be answered, they could be improved or augmented to cover a wider range of questions. This section briefly considers a number of directions that continued research could take to further these approaches to factoid and definition questions. For future directions in QA, not directly related to the work in this thesis, interested readers should see Maybury (2004).

13.4.1 User Centred Evaluations

All of the evaluations detailed in this thesis have focused solely on the performance of the underlying technology. It is important to remember, however, that the long term goal of question answering research is to develop systems which allow average computer users easy and convenient access to the vast amounts of information available as free text. To fulfil this aim due care and consideration must be given to the construction of question answering applications, i.e. the interface through which users will interact with the underlying QA technology.

Whilst interfaces for information retrieval have been extensively studied (for a good overview of IR interfaces and their evaluation see Baeza-Yates and Ribeiro-Neto (1999)) the same cannot be said for question answering systems. For example, whilst this thesis has introduced two QA systems, AnswerFinder and Varro (see Chapter 9 and Appendix C respectively), the interfaces were not the focus of this thesis work and were designed purely to please the author. Different interfaces display answers in different ways with differing levels of detail and as such can affect user centered evaluations and should be investigated further.

Lin et al. (2003) performed the only known user centred evaluation of a QA system. Their evaluation, however, concentrated only on the amount of context that should be returned with a correct answer and did not evaluate different interfaces or how the system should respond if for instance no answer can be located.

For question answering to really move forward and become a widely used tool comprehensive user testing has to be undertaken to evaluate numerous aspects of their design including but not limited to: how many answers should be displayed, how much context should be given with each answer, how should failure to find answers be presented.

13.4.2 Information Retrieval Techniques

A number of the approaches to question answering introduced in this thesis have made use of techniques from the information retrieval community. The Lucene IR engine was used for retrieving text in both the factoid and definition systems (Chapters 7 and 11 respectively) and IR techniques were also used for question classification and sentence similarity (Sections 6.1.2 and 12.1 respectively).

All the retrieval approaches have used a standard vector space ranking model (albeit with a minor modification for the `alt` operator introduced in Section 6.2.2) and term weighting has been determined using the basic TF.IDF model. While this approach has performed acceptably, it is possible that other approaches would provide higher performance. A number of term weighting schemes have been described which outperform the basic TF.IDF weighting on specific collections. It is unclear, however, if these alternative term weighting schemes would also show improvements when evaluated in the context of QA. For example, Section 7.1 of this thesis makes reference to the fact that Lucene using TF.IDF and Okapi using BM25 (Robertson et al., 1994) have similar coverage performance.

A number of approaches to recall enhancement, either by query expansion or refinement, have been introduced and evaluated in this thesis (Section 6.2 and Chapter 11). More traditional approaches, such as relevance feedback, have not been used even though they may be able to contribute to improved IR performance within the context of QA.

An interesting approach to experimenting with both multiple term weighting models and recall enhancement techniques would be to adapt the techniques introduced in this thesis

to use the Terrier[1] IR engine (Ounis et al., 2005) in place of Lucene. Terrier provides support for multiple retrieval models including TF.IDF, BM25, and Ponte-Croft language modelling(Ponte and Croft, 1998) and a number of recall enhancement techniques. Using Terrier would therefore allow experimentation with different techniques whilst retaining the same underlying document index allowing for a fair comparison of the different approaches.

13.4.3 More Complex Answer Models

During the discussion in Chapter 8 of our approaches for extracting answers to factoid questions we suggested that intelligent tokenization of dates and numbers would allow more accurate comparison of multiple answers. This approach could be extended to compare other semantic types, further improving the comparison of answers. For example, measurements could be converted to a common unit and then compared. Distance measurements could all be converted to meters while weight measurements could be converted to grams. Determining if extending the answer comparison in this way improves performance would be relatively straightforward as we could make use of the Google calculator[2]. Not only does the Google calculator allow simple mathematical expressions to be evaluated but it can also perform conversions between different measurement units and so is ideally suited to the task. In any real-world application, however, the comparisons would have to be carried out much more quickly than would be possible using a website as part of the tokenization or answer comparison algorithm.

As well as considering the equivalence of answers, other relations between candidate answers could be considered. For example, Dalmas and Webber (2004) also consider the inclusion of candidate answers when processing *where* questions. Given a set of all possible answer entities a complex answer model can be constructed. In the reported experiments WordNet was used to determine the inclusion relations. An answer model, showing all the candidate answer instances found, for the question *"Where is Glasgow?"* can be seen in Figure 13.1.

Using just equivalence between possible candidate answer instances as a way of selecting an answer would result in London being returned as it is the only answer candidate to appear twice. Taking into account inclusion would allow a system to determine that Glasgow is in Scotland which in turn is in Britain. This type of answer model allows for different answers to the same question to be returned (both Scotland and Britain are correct answers) but more interestingly allows for answer generation. Instead of returning a single entity, as found in a relevant document, the system could in fact return a combined answer such as "Scotland, Britain".

Such an approach to inclusion could be incorporated in the semantic type answer extraction component of Section 8.1. Of course there is no reason that inclusion should be

[1] Available from `http://ir.dcs.gla.ac.uk/terrier/`

[2] `http://www.google.com/help/calculator.html`

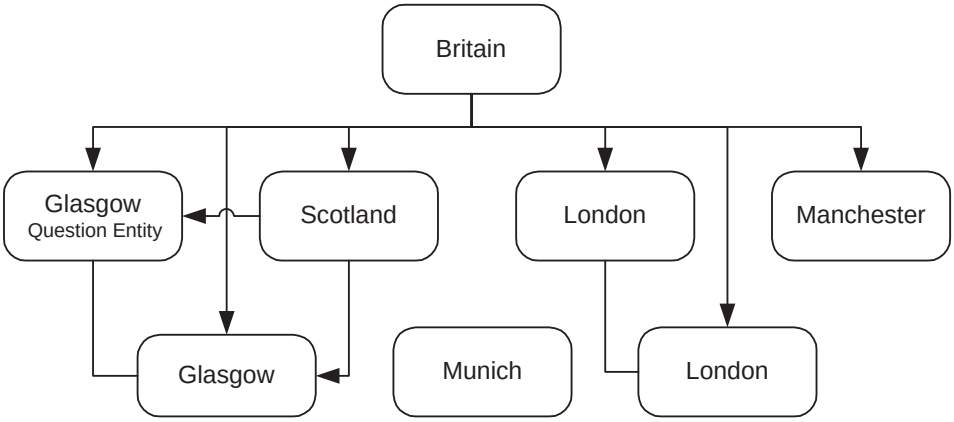

Figure 13.1: Answer model for *"Where is Glasgow?"* (arrows stand for *inclusion* and lines *equivalence*).

limited to locations. For instance, dates could be handle in a similar fashion. If the answer candidates "23rd of July 1980", "July 1980", and "1980" were extracted for a question then while they could not be classed as equivalent, they can be related via inclusion.

The end of Chapter 8 discussed a number of limitations of our approaches to answer extraction. One of these limitations was the inability to recognise when a question did not have an answer. Inclusion may also provide a way of determining that there is no correct answer, or at least that none of the candidate answers which the system has found are correct. The system could decide that there is no correct answer if the question entity cannot be incorporated into the answer model, i.e. if it is not included or does not itself include any other node in the model.

13.4.4 Mixed Media Answers

One of the main motivations for the work in this thesis was that with the continued growth in the amount of unstructured information new more efficient access methods were needed to fully utilize this large store of knowledge. While for many years web search engines have been an appropriate way of accessing the Internet, it is more and more the case that there is simply too much information for these techniques to continue to be the most appropriate way of searching the web.

To date, most of the research into question answering has revolved around providing textual answers to a question. As a replacement for web search engines this is an ideal step as people are used to receiving text in response to web searches (many people already enter natural language questions as web search queries so for them the switch to using QA technology should be easy and instantly beneficial). The answers to many questions, however, can be realised just as well, if not better through other mediums.

Just as web search engines have moved into indexing content other than text (the Google Image search is a good example[3]) QA research is also beginning to focus on returning an-

[3] `http://images.google.com`

swers using varied media formats. For example, asking *"What is Hawaii's state flower?"* should probably result in returning the phrase *"yellow hibiscus"* as well as a picture of the flower.

AnswerFinder (see Chapter 9) makes an initial attempt at returning answer elaboration in formats other than free text. For example, if the answer to a question is a location then (where possible) a link to a map showing the answer will also be provided. This approach, however, is limited in its scope as it relies on existing knowledge sources which can be easily searched for items related to answers of a specific semantic type.

Extending QA systems to return answers which are more complex than a simple text snippet would be beneficial to the users of such systems. Having such complex answers would also allow QA systems to answer a wider range of questions as well as providing an easier way of answering certain question types. For example, using just text to answer the question *"What did Abraham Lincoln look like?"* would require a description of a few sentences in length, whereas systems which can make use of other media formats could answer this question by displaying a portrait of Lincoln. Questions such as *"Who was Benjamin Franklin?"* could be answered by displaying a biography, a portrait, links to maps of places he is associated with (where he was born, lived, is buried, etc.) and anything else the system deems to be relevant.

One such system is the Wee QA system (Ahn et al., 2004) in combination with an image based renderer (Dalmas et al., 2004) developed during the IGK 2004 Summer School. Wee returns an illustrated summary of each different answer topic it locates for a question, where an illustrated summary is a text snippet and one or more relevant pictures.

13.5 Summary

This thesis has introduced a number of novel approaches to answering factoid and definition questions. While the approaches do not outperform the current state-of-the-art systems, when combined they result in a system which scores above average when independently evaluated within the TREC framework (the system performs above average on all three question types: factoid, list, and definition – see Appendix E for details).

The motivation behind this thesis was not to produce a state-of-the-art QA system but rather to investigate techniques which could be easily understood and also be used to create QA systems capable of quickly answering questions. It is, therefore, encouraging that not only are the techniques introduced in this thesis capable of answering questions relatively quickly but that their answer performance is above average when independently evaluated.

It is hoped that the ideas presented in this thesis will inspire others to develop further techniques for QA that will be even more effective than the ideas in this thesis alone and will ultimately lead to the creation of high speed, high accuracy question answering systems.

Appendix A

How Many Books?

One of the problems encountered when dealing with text collections is envisioning exactly how vast the collections can be. Often the size of a collection is given in gigabytes and it can be difficult to visualise this in terms of printed words which we are all used to dealing with. This appendix gives equivalences for a number of storage sizes to allow the reader to properly comprehend the size of collections such as AQUAINT or the web[1].

- Byte
  - 1 byte: a single character
- Kilobyte (KB): 10^3 bytes
  - 1 KB: A very short story
  - 2 KB: A typewritten page
- Megabyte (MB): 10^6 bytes
  - 1 MB: A small novel
  - 5 MB: The complete works of Shakespeare
  - 100 MB: A metre of shelved books
- Gigabyte (GB): 10^9 bytes
  - 1 GB: A pickup truck filled with books
  - 100 GB: A library floor of academic journals
- Terabyte (TB): 10^{12} bytes
  - 1 TB: 50,000 trees made into paper and printed.
  - 2 TB: An academic research library
  - 10 TB: The print collections of the U.S. Library of Congress

Given these figures we can see that the Cystsic Fibrosis Database (Shaw et al., 1991) at 5 MB is equivalent to the complete works of Shakespeare, the AQUAINT collection at approximately 3 GB is equivalent to three pickup trucks filled with books and the world wide web at approximately 167 TB (Lyman and Varian, 2003) is the equivalent of nearly 17 copies of every item in the print collections of the U.S. Library of Congress.

[1] These figures originated in (Williams, Mid 1990s) which, unfortunately, is no longer available.

Appendix B

Small Web Based Question Set

For a number of experiments a small web based question set was required and the decision was taken not to use questions which have been previously used as part of a TREC QA evaluation. This decision was taken after it had been noted that the TREC questions are now appearing quite frequently (sometimes with correct answers) in the results of web search engines. This could have affected the results of any web based study. For this reason a new collection of fifty questions was assembled to serve as the test set.

The questions within the new test set were chosen to meet the following criteria:

1. Each question should be an unambiguous factoid question with only one known answer. Some of the questions chosen do have multiple answers although this is mainly due to incorrect answers appearing in some web documents.

2. The answers to the questions should not be dependent upon the time at which the question is asked. This explicitly excludes questions such as *"Who is the President of the US?"*.

We include the full test set here to allow other interested parties to test systems over the same question sets, although we ask that these questions not be placed in documents indexed by web search engines, as this would defeat the purpose of the test set.

Q001: The chihuahua dog derives it's name from a town in which country?
Mexico

Q002: What is the largest planet in our Solar System?
Jupiter

Q003: In which country does the wild dog, the dingo, live?
Australia or America

Q004: Where would you find budgerigars in their natural habitat?
Australia

Q005: How many stomachs does a cow have?
Four or one with four parts

Q006: How many legs does a lobster have?
Ten

Q007: Charon is the only satellite of which planet in the solar system?
Pluto

Q008: Which scientist was born in Germany in 1879, became a Swiss citizen in 1901 and later became a US citizen in 1940?
Albert Einstein

Q009: Who shared a Nobel prize in 1945 for his discovery of the antibiotic penicillin?
Alexander Fleming, Howard Florey or Ernst Chain

Q010: Who invented penicillin in 1928?
Sir Alexander Fleming

Q011: How often does Haley's comet appear?
Every 76 years or every 75 years

Q012: How many teeth make up a full adult set?
32

Q013: In degrees centigrade, what is the average human body temperature?
37, 38 or 37.98

Q014: Who discovered gravitation and invented calculus?
Isaac Newton

Q015: Approximately what percentage of the human body is water?
80%, 66%, 60 percent or 70 percent

Q016: What is the sixth planet from the Sun in the Solar System?
Saturn

Q017: How many carats are there in pure gold?
24

Q018: How many canine teeth does a human have?
Four

Q019: In which year was the US space station Skylab launched?
1973

Q020: How many noble gases are there?
6

Q021: What is the normal colour of sulphur?
Yellow

Q022; Who performed the first human heart transplant?
Dr Christiaan Barnard

Q023: Callisto, Europa, Ganymede and Io are 4 of the 16 moons of which planet?
Jupiter

Q024: Which planet was discovered in 1930 and has only one known satellite called Charon?
Pluto

Q025: How many satellites does the planet Uranus have?
15, 17, 18 or 21

Q026: In computing, if a byte is 8 bits, how many bits is a nibble?
4

Q027: What colour is cobalt?
blue

Q028: Who became the first American to orbit the Earth in 1962 and returned to Space in 1997?
John Glenn

Q029: Who invented the light bulb?
Thomas Edison

Q030: How many species of elephant are there in the world?
2

Q031: In 1980 which electronics company demonstrated its latest invention, the compact disc?
Philips

Q032: Who invented the television?
John Logie Baird

Q033: Which famous British author wrote ”Chitty Chitty Bang Bang”?
Ian Fleming

Q034: Who was the first President of America?
George Washington

Q035: When was Adolf Hitler born?
1889

Q036: In what year did Adolf Hitler commit suicide?
1945

Q037: Who did Jimmy Carter succeed as President of the United States?
Gerald Ford

Q038: For how many years did the Jurassic period last?
180 million, 195 – 140 million years ago, 208 to 146 million years ago, 205 to 140 million years ago, 205 to 141 million years ago or 205 million years ago to 145 million years ago

Q039: Who was President of the USA from 1963 to 1969?
Lyndon B Johnson

Q040: Who was British Prime Minister from 1974-1976?
Harold Wilson

Q041: Who was British Prime Minister from 1955 to 1957?
Anthony Eden

Q042: What year saw the first flying bombs drop on London?
1944

Q043: In what year was Nelson Mandela imprisoned for life?
1964

Q044: In what year was London due to host the Olympic Games, but couldn’t because of

the Second World War?
1944

Q045: In which year did colour TV transmissions begin in Britain?
1969

Q046: For how many days were US TV commercials dropped after President Kennedy's death as a mark of respect?
4

Q047: What nationality was the architect Robert Adam?
Scottish

Q048: What nationality was the inventor Thomas Edison?
American

Q049: In which country did the dance the fandango originate?
Spain

Q050: By what nickname was criminal Albert De Salvo better known?
The Boston Strangler

Appendix C

Defining Terms with Varro

AnswerFinder (see Chapter 9) was developed as a simple interface to many of the approaches to factoid question answering detailed in Part II of this thesis. In a similar way Varro[1] was designed to showcase and make publicly available the approaches to answering definition questions detailed in Part III of this thesis. We have described just one approach to definition question answering, unlike the multiple techniques which were developed for factoid QA, and this is the approach implemented in Varro[2].

Traditionally, users wanting a concise definition of an entity or event would consult either a dictionary or encyclopaedia. With the growth of the home computing market in the early 1990s many people switched from paper based encyclopaedias to electronic versions such as Microsoft's Encarta. As technology advances with the rapid growth of the Internet more people are ignoring encyclopaedias altogether and turning straight to the web using search engines, such as Google, to find the required information. This information is usually freely available unlike the hefty and expensive paper based reference sets of the 1980s or even the cheaper more accessible CD-ROM editions, such as Microsoft's Encarta[3].

The main problem with using a web search engine to try to define an entity or event is that the user may have to read a number of articles from different sites to build up a detailed description which can be a time consuming exercise. Also all the information about the target will have to be copied from the pages to create a single definition; a task which may well be time consuming and certainly is not necessary when using an encyclopaedia as all the information about a given target is presented together in a single location. The obvious advantages to using the web as a source of information is the wide range of topics that are covered and the fact that it is being constantly added to both by reputable news agencies (for the definition of current events and figures) and also by people interested in a wide variety of topics almost guaranteeing that some information will exist for any target a user wishes to define.

[1] Marcus Terentius Varro (116-27 BC) was a Roman scholar and the first Roman encyclopaedist. He compiled an encyclopaedia of the liberal arts entitled *Disciplinae* (The Disciplines, 30 BC).

[2] Freely available from `http://www.dcs.shef.ac.uk/~mark/phd/software/`

[3] `http://www.microsoft.com/products/encarta/`

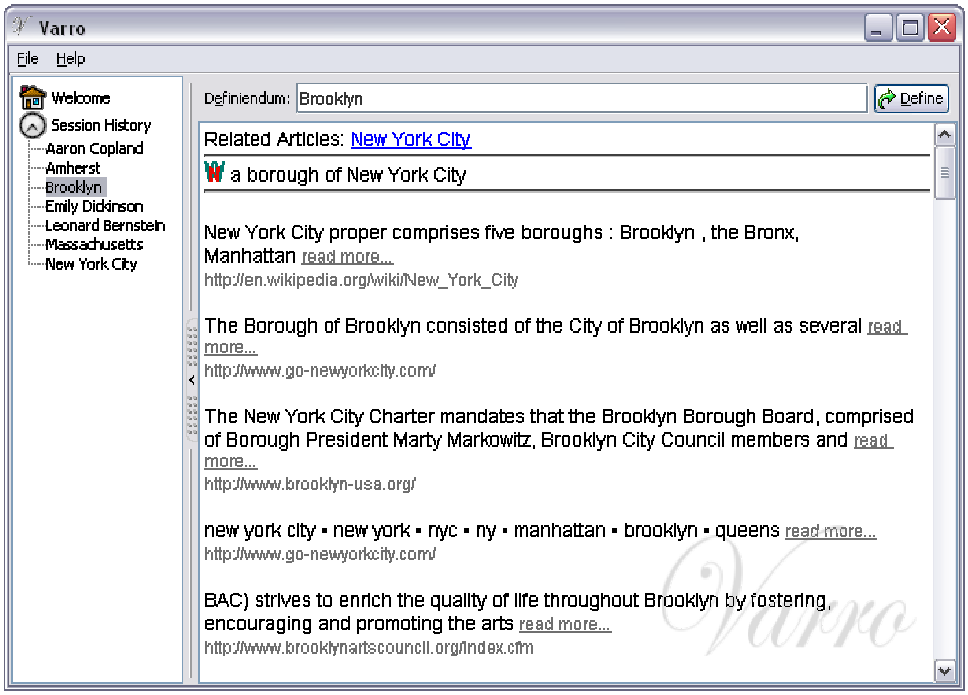

Figure C.1: Varro: a web based open-domain definition system.

User Interface

An ideal application would therefore be an encyclopaedia which instead of having pre-defined entries for a fixed number of targets would scour the Internet to construct an up-to-date definition of any target it was given. This was the motivation behind the development of Varro, the user interface of which can be seen in Figure C.1.

The interface consists of two main areas, the result display which contains: links to related definitions, the WordNet entry for the target (if available) and the definition composed from relevant documents located via Google using the approach detailed in the previous chapters. The other part of the interface currently lists previous searches to enable the user to go back and re-read a definition.

On average Varro takes approximately 21 seconds to construct a new definition from the web[4]. Whilst this is a relatively long time compared to modern web search engines we must take a number of things into account. Firstly Varro is performing a much more complex function than a search engine – producing a definition. Secondly and more importantly no work was carried out to optimise Varro for speed which would probably have a dramatic impact on the performance. Given this 21 seconds does not seem an overly long time in which to construct a definition.

[4] When run on a PC with a 3GHz Pentium 4 CPU and 1Gb of RAM

Comparison with other Web Based Definition Question Answering Systems

As automatic definition creation is a relatively new research field, there is little in the way of publicly available systems which use the Internet as their knowledge base. Two systems that are known to the author are Google's *define* keyword and DefSearch.

The most accessible definition system is the Google web search engine. Users can get multiple definition sentences simply by entering *define: word* as a query into Google's search interface, where word can be any target you wish to define. For example *define: golden parachute* will give a number of sentences which define the term 'golden parachute' as among other things 'compensation paid to top-level management by a target firm if a takeover occurs'. While providing many relevant sentences for a given target it is unclear if any attempt has been made to remove redundancy and present a concise definition. Many of the retrieved sentences relay the same information. Very little information is available about the implementation aspects of the system although it is clear that definitions are often drawn from the online version of WordNet[5], as well as sites which contain the words *definition* or *glossary* within their URL. Unfortunately while Google seems to be quite good at providing definitions for inanimate objects or events it tends not to provide any definitions for well known people.

Another online definition QA system is DefSearch[6]. This appears to be a development of the definition component of the QUALIFIER system, entered in the TREC 2003 QA evaluation (Yang et al., 2003), via the addition of hard and soft matching rules (Cui et al., 2004a; Cui et al., 2004b). Unfortunately the current publicly accessible version is not capable of producing answers in real-time – the systems FAQ suggests that it takes between ten and twenty minutes to produce an answer[7]. This is a shame as the underlying system achieved an F(β=5) score of 0.473 in the TREC 2003 QA evaluation (Yang et al., 2003) and later work (Cui et al., 2004a) suggests the most recent version of the system can achieve an F(β=5) score of 0.539 over the same question set.

[5] `http://www.cogsci.princeton.edu/cgi-bin/webwn/`

[6] `http://www-appn.comp.nus.edu.sg/~cuihang/DefSearch/DefSearch.htm`

[7] In the authors experience it can actually take a lot longer than the stated twenty minutes for a definition to be fully generated.

Appendix D

Official TREC 2003 Results

A system comprised of early prototype versions of the semantic type extraction and surface matching text patterns (see Sections 8.1 and 8.2 respectively) using the top 20 paragraphs retrieved from AQUAINT by Okapi, was independently evaluated as part of the TREC 2003 question answering evaluation. This combined system is referred to as `shef12simple` in the TREC reports (Gaizauskas et al., 2003; Greenwood and Saggion, 2004). The official results for the factoid and list components of the evaluation are given below[1].

Factoid Questions

The factoid component of the evaluation consisted of 413 questions. The official published statistics for the submission were:

Number wrong (W): 338
Number unsupported (U): 12
Number inexact (X): 6
Number right (R): 57
Accuracy: 0.138
Precision of recognizing no answer: 15 / 191 = 0.079
Recall of recognizing no answer: 15 / 30 = 0.500

Figure D.1 shows how these results compare with those of the other runs submitted as part of the evaluation.

Of the 413 questions 12 were suitable for answering with the surface matching text patterns described in Section 8.2, unfortunately none of the patterns selected any answers, correct or otherwise. This is probably due to the difference in the style of language used

[1] The definition component of the entry was kindly provided by Horacio Saggion, as the definition system described in Part III was not developed in time for the TREC 2003 QA evaluation.

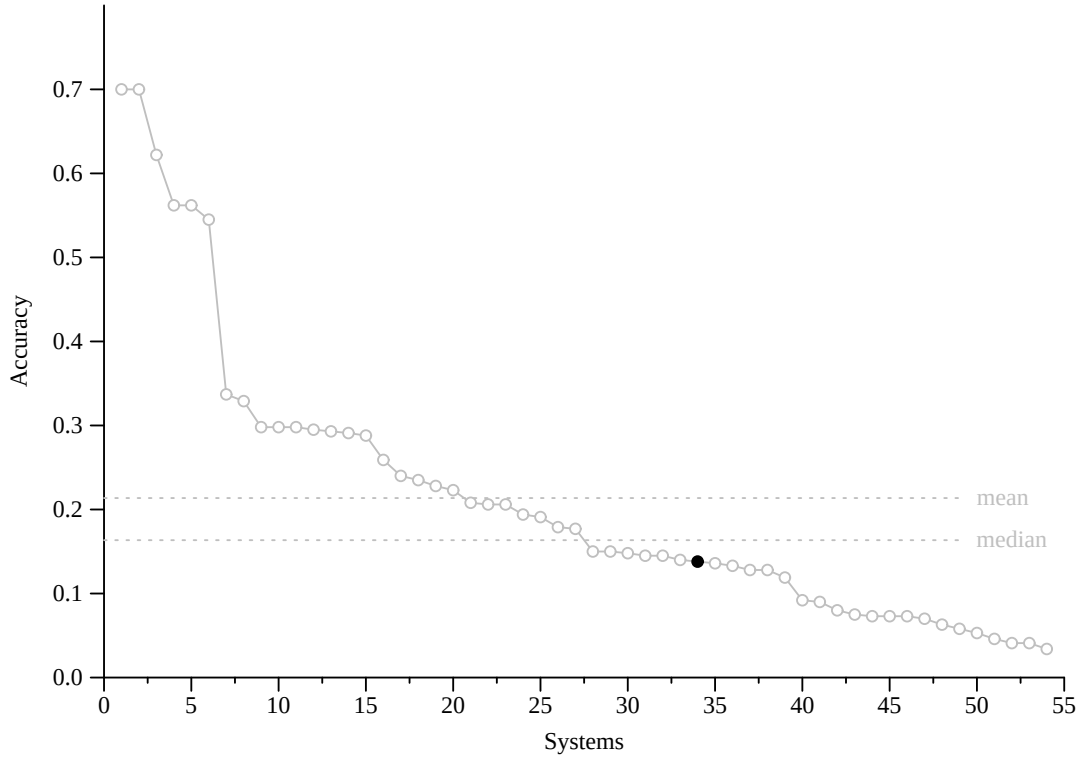

Figure D.1: Official TREC 2003 factoid scores (• is `shef12simple`).

		Correct	Incorrect	Total
Question Analysis	Assigned UNKNOWN	119	27	146
	Assigned known type	241	26	267
	Total	360	53	413
Answer Extraction	Returned NIL	15	176	191
	Returned an answer	42	180	222
	Total	57	356	413

Table D.1: Summary of TREC 2003 factoid performance.

in web documents, over which the patterns were acquired, compared to newswire articles from which we attempted to extract answers (see Greenwood and Saggion (2004) for a fuller discussion of these failings). The semantic type extraction system assigned an incorrect type to 53 questions, 27 of these were typed as UNKNOWN (i.e. the answer extraction component will make no attempt to answer them) leaving 26 questions for which the system could only return answers of an incorrect semantic type. In total 146 of the questions were assigned the UNKNOWN type, so 267 questions were typed, 241 correctly. The system returned NIL for 191 of the questions so the system was unable to find an answer for 45 of the typed questions. Of the remaining 241 questions, 18 have no known answer leaving 223 questions to which the system could return a correct non-NIL answer. This is summarised in Table D.1.

Unfortunately Okapi was only able to locate answer bearing passages for 131 of the 223 questions correctly assigned a type. This means that the maximum obtainable score for the whole system is 0.317 (131/413), however, the official TREC score was 0.138 (57/413)

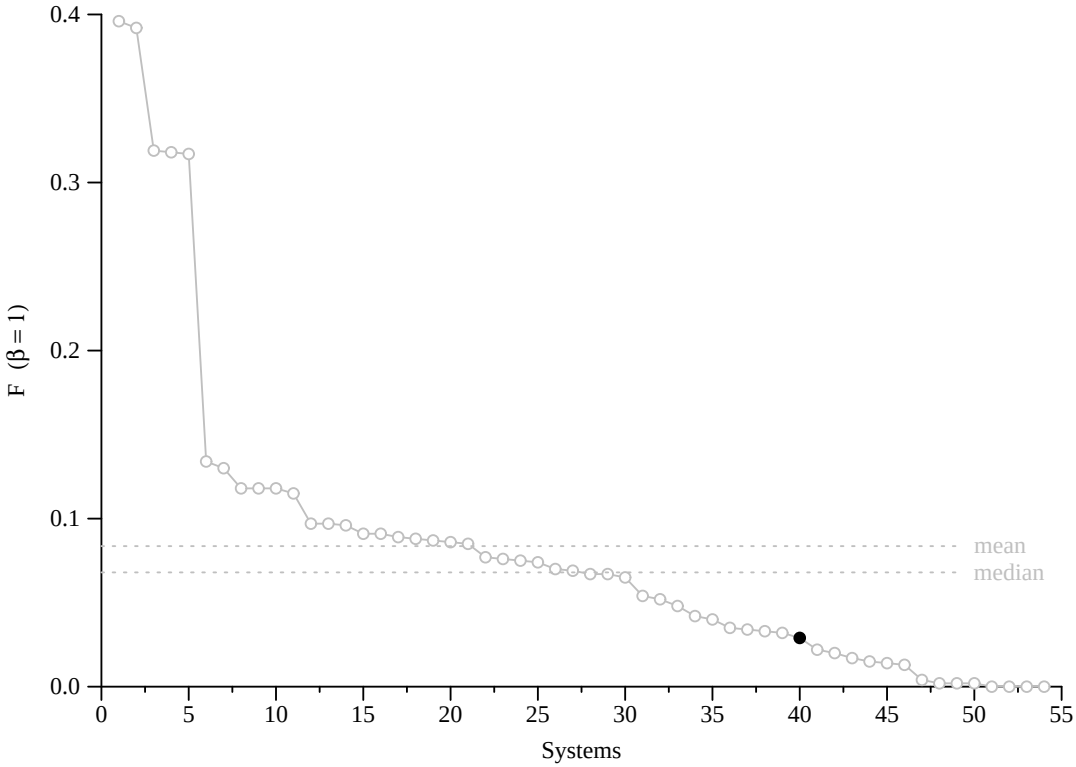

Figure D.2: Official TREC 2003 list scores (● is `shef12simple`).

which contains fifteen correct NIL responses and so the system actually provided correct supported non-NIL exact answers for 42 questions giving a score of 0.102 which is 32.2% of the maximum attainable score.

List Questions

The official average F(β=1) score obtained by the system over the 37 list questions was 0.029. Figure D.2 shows how this result compares with those of the other runs submitted as part of the evaluation.

The system encountered similar problems when answering the list questions as when answering the factoid questions as well as a problem specific to list questions. Over the 37 questions the system returned 20 distinct correct answers achieving the F(β=1) score of 0.029. Unfortunately the ability of the system to locate a reasonable number of distinct answers was offset by the fact that for each question many answers (i.e. all those found) are proposed, dramatically lowering the precision and hence the F(β=1) score. For example, within the AQUAINT collection, there are seven known answers to the question *"What countries have won the men's World Cup for soccer?"* for which the system returned 32 answers only two of which were correct giving a recall of 0.286 but a precision of only 0.062.

Appendix E

Official TREC 2004 Results

The system entered in the QA track of TREC 2004 consisted of the semantic type extraction and surface matching text patterns (see Sections 8.1 and 8.2) as well as the WordNet based extension to the semantic type extraction system detailed in Section 8.1.2. The top 20 passages were retrieved from AQUAINT using the open-source Lucene IR engine. This combined system, referred to as `shef04afv`, is detailed in Gaizauskas et al. (2004).

The format of the 2004 TREC QA evaluation differs from previous years in that a number of questions are asked about a given target (i.e. there is some notion of context) – see Voorhees (2004) for an overview of the task and the performance of other participants. An example scenario for target 3, *"Hale Bopp comet"* is:

Q3.1 (Factoid): When was the comet discovered?
Q3.2 (Factoid): How often does it approach the earth?
Q3.3 (List): In what countries was the comet visible on its last return?
Q3.4 (Definition): Other

This revised question format caused some problems for the systems detailed in this thesis as the systems described are designed to answer a single unambiguous question whereas the new question format requires that a question be considered in relation to a target, other questions and possibly their answers. The two main problems encountered with the question format were:

- In many questions rather than the actual target a pronoun appears in it's place. The system compensated for this by appending the target to the end of the question if it was not already present. This was ample manipulation of the question for the IR component to retrieve relevant documents and in most cases also allowed the question analysis component to correctly type the question. Unfortunately the lower level type rules will type the question as Person (male or female) based only on the presence of a pronoun (he or she respectively). For instance target 4 is *James*

Dean and Q4.4 *"Which was the first movie that he was in?"* is typed as Person of type male purely because of the presence of the pronoun *he*, whereas if it is typed at all it should be of type 'movie' (whatever that relates to in the underlying question classification set or hierarchy).

- Each of the 65 targets had a definition question associated with it (also referred to as questions of type 'other'). The main problem with these questions stems from the fact that the guidelines state that correct answers to the factoid and list questions for a target will not be counted as valid nuggets for the evaluation of the definition question. Brief testing during system development showed that the system struggled to remove sentences containing answers to the factoid and list questions without also removing other valid nuggets and hence decreasing the performance of the system. For this reason no attempt was made to remove answers to the factoid and list questions from the answers to the definition questions.

It has been mentioned throughout this thesis that the long term goal of open-domain QA is to provide systems which can correctly answer questions in real-time, i.e. as a possible replacement for modern day web search engines. This QA system answered all 351 questions in roughly 75 minutes which equates to approximately 13 seconds per question[1] – a not unreasonable length of time to have to wait for an answer.

The following sections detail the results for the three different question types. It is interesting to note that all three results are above the median (and mean) of the submitted systems, which is a definite improvement over the prototype systems evaluated as part of TREC 2003 (see Appendix D) for which the results were all below the median. Of the 28 participating groups the `shef04afv` run was ranked 9^{th} (when all three question types were combined) with it ranking 18^{th} out of the 63 submitted runs.

Factoid Questions

The factoid component of the evaluation consisted of 230 questions. The official published statistics for the run were:

Number wrong (W): 170
Number unsupported (U): 4
Number inexact (X): 7
Number right (R): 49
Accuracy: 0.213
Precision of recognizing no answer: 3 / 42 = 0.071
Recall of recognizing no answer: 3 / 22 = 0.136

Figure E.1 shows how these results compare with those of the other runs submitted as part of the evaluation.

[1] The PC on which the run was performed was a 2.2GHz Pentium 4 with 512Mb of memory.

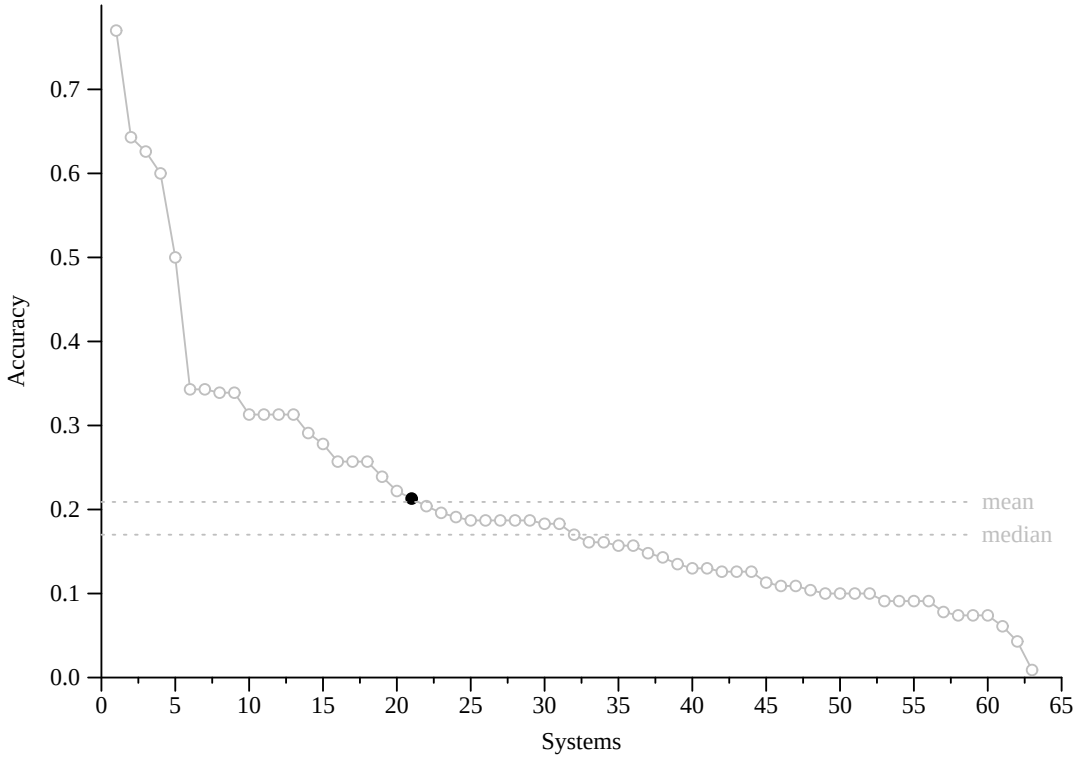

Figure E.1: Official TREC 2004 factoid scores (● is `shef04afv`).

		Correct	Incorrect	Total
Question Analysis	Assigned UNKNOWN	27	5	32
	Assigned known type	177	21	198
	Total	204	26	230
Answer Extraction	Returned NIL	3	39	42
	Surface Matching Patterns	0	1	1
	Semantic Type Extraction	44	134	178
	WordNet Extension	2	7	9
	Total	49	181	230

Table E.1: Summary of TREC 2004 factoid performance.

Question analysis was such that of the 230 factoid questions: 1 was answered by the surface matching text patterns, 188 by the semantic type extraction component, and 9 by the WordNet based extension to the semantic type extraction system. The remaining 32 questions were assigned the UNKNOWN type. Unfortunately 21 of the 188 questions answered by the semantic type extraction system were incorrectly typed reducing the maximum number of questions which could be correctly answered to 177 (i.e. a maximum attainable score of 0.770). This analysis is summarised in Table E.1.

Given the way the system was able to classify the questions the official score of 0.213 is approximately 30% of the maximum attainable score although 3 of the correct answers were NIL responses returned as the answer type could not be determined from the question and so the system actually returned correct non-NIL answer for 46 questions giving a score of 0.200 – 26.0% of the maximum attainable score.

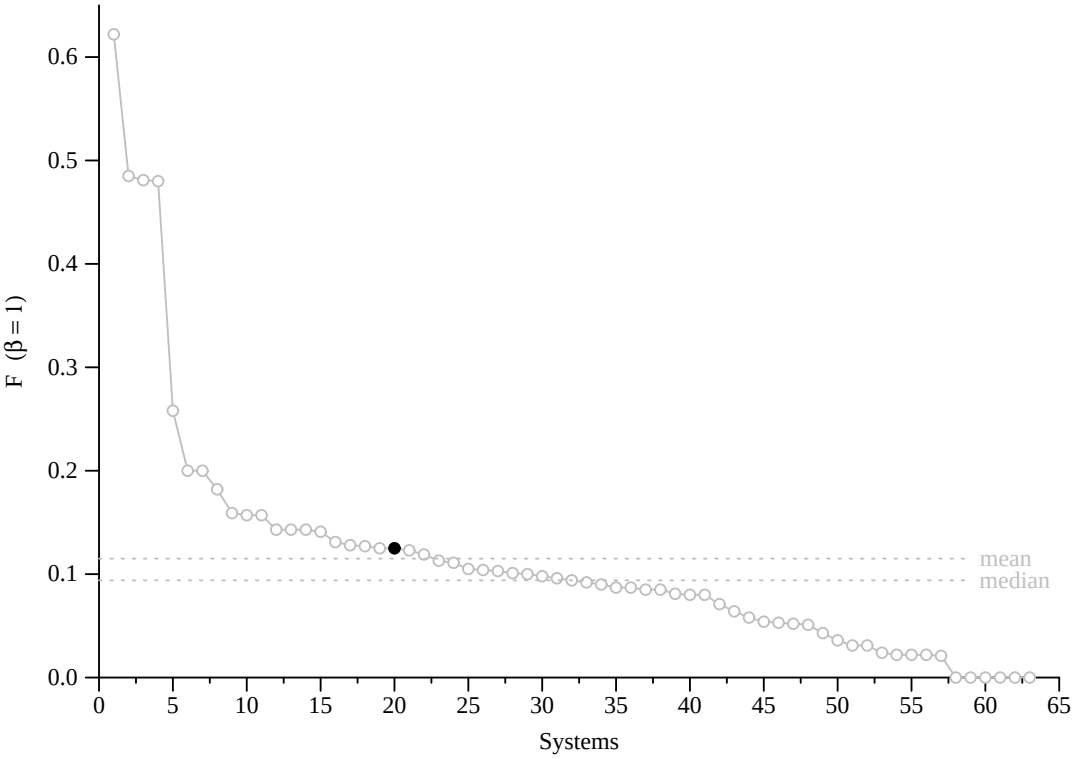

Figure E.2: Official TREC 2004 list scores (● is `shef04afv`).

List Questions

The official average F(β=1) score over 55 list questions[2] in the evaluation was 0.125. Figure E.2 shows how this result compares with the other submitted runs.

Of the 55 list questions 44 were answered by the semantic type extraction system (2 of which were answered by the WordNet extension). Unfortunately 9 of these questions were assigned the wrong answer type and so could not be successfully answered. No answers were given for the remaining 11 questions which were of an UNKNOWN type[3].

The main reason behind the improvement of the system compared to the performance in the TREC 2003 evaluation is the way in which the system determines the number of answers to return for each question. For the TREC 2003 evaluation the system simply used the factoid QA system to locate all possible answers to a question all of which were then returned as answer to the list question. Unfortunately while this approach is capable of finding correct answers to the list questions it also retrieves many wrong answers to each question which swamps the correct answers resulting in a relatively high recall but an exceptionally low precision.

[2] The original question set contained 56 questions, however, Q25.4 was discarded as it has no known answers in the AQUAINT collection which contravenes the evaluation guidelines.

[3] The evaluation guidelines do not allow a system to return a NIL response for a list question, unlike with the factoid questions, and so when the system does not know how to answer a list question the system response is UNKNOWN with the doc ID APW19980601.0003.

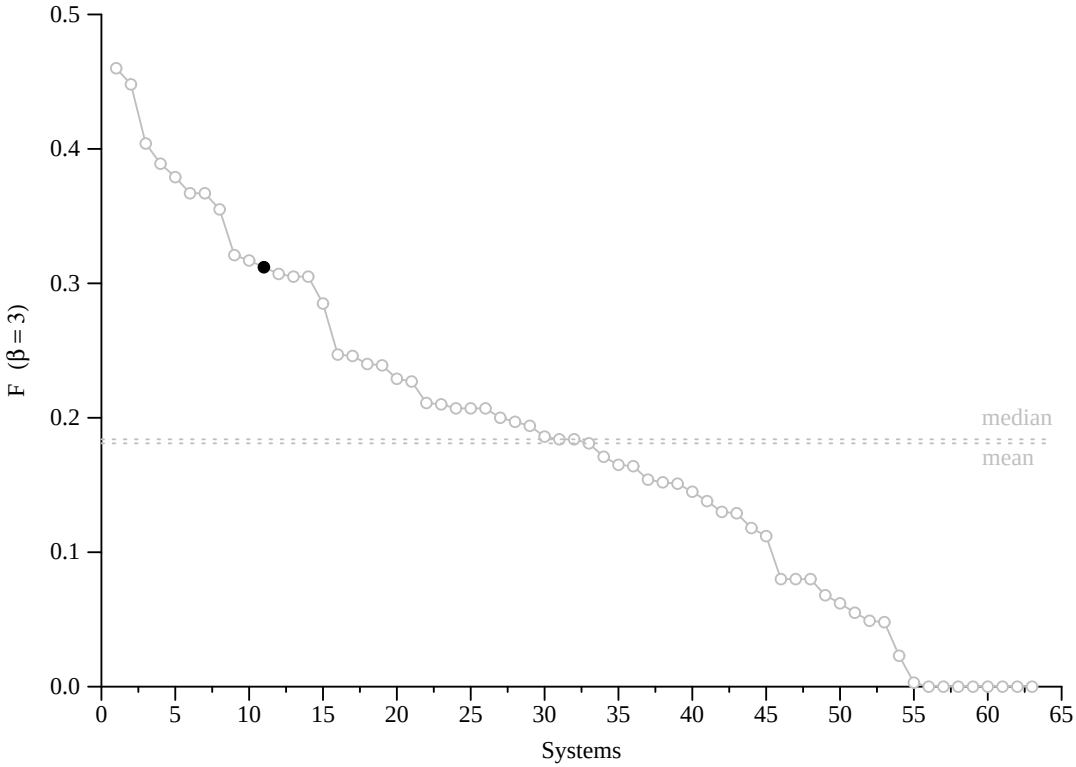

Figure E.3: Official TREC 2004 definition scores (• is `shef04afv`).

For the TREC 2004 evaluation a more sensible approach was adopted to limit the number of answers returned per question without reducing the number of correct answers returned. If the system finds 10 answers or less for a question then they are all returned. On the other hand if more than 10 answers are located then the first 10 are returned along with any further answers which have a confidence score of 0.08 or above (calculated using Equation 8.1).

Over the 55 list questions this more refined approach to answer selection resulted in the locating of 54 distinct correct answers to give the F(β=1) score of 0.125 while returning on average 8.5 answers per question with at least one correct answer (down from the average per question with at least one correct answer of 20.75 for the TREC 2003 evaluation).

Definitions

The official average F(β=3) score for the 64 definition questions[4] used in the evaluation was 0.312. Figure E.3 shows how this result compares with those of the other runs submitted for evaluation.

Of the 64 definition questions, which were evaluated, the system created a definition for all but two questions. Unfortunately of the 62 definitions the system constructed only 44

[4] There were originally 65 questions, however, Q7.4 was never judged – no reason has yet been given by NIST as to why the question was discarded.

of them contained at least one nugget deemed vital by the assessors although most of the other 18 definitions did contain one or more acceptable nuggets but were never the less awarded a score of 0 (see Section 3.1.3 for a full description of the evaluation metric).

Appendix F

Official TREC 2005 Results

The system entered in the QA track of TREC 2005 consisted of improvements to the semantic type extraction and surface matching text patterns (see Sections 8.1 and 8.2), the WordNet based extension to the semantic type extraction system detailed in Section 8.1.2, and a simple noun chunk system, detailed later in this appendix, for boosting performance when answering list questions. The top 20 passages were retrieved from AQUAINT using the open-source Lucene IR engine. This combined system is referred to as `shef05lmg` in the description by Gaizauskas et al. (2005a).

The format of the TREC 2005 QA evaluation follows the format of the 2004 evaluation (see Appendix E), the only difference being that the targets can now be events as well as entities. For example the target of question set 66 was *"Russian submarine Kursk sinks"* about which the following questions were asked:

Q66.1 (Factoid): When did the submarine sink?
Q66.2 (Factoid): Who was the on-board commander of the submarine?
Q66.3 (Factoid): The submarine was part of which Russian fleet?
Q66.4 (Factoid): How many crewmen were lost in the disaster?
Q66.5 (List): Which countries expressed regret about the loss?
Q66.6 (Factoid): In what sea did the submarine sink?
Q66.7 (List): Which U.S. submarines were reportedly in the area?
Q66.8 (Definition): Other

Allowing targets to be event descriptions (rather than just named entities, e.g. *"World War II"*) makes it more difficult to modify the questions to include the target as the targets are less structured and harder to process.

For this evaluation the approach we used to combine questions and targets in the TREC 2004 evaluation was extended to use both pronominal and nominal coreference resolution. For example, consider the seven questions for target 75, *"Merck & Co.*", for which the processed questions are shown in Figure F.1. Note that this approach does not always

Question Number	Original and Modified Question
75.1	Where is the company headquartered? Where is Merck & Co. headquartered?
75.2	What does the company make? What does Merck & Co. make?
75.3	What is their symbol on the New York Stock Exchange? What is Merck & Co.'s symbol on the New York Stock Exchange?
75.4	What is the company's web address? What is Merck & Co.'s web address?
75.5	Name companies that are business competitors. Name companies that are business competitors.
75.6	Who was a chairman of the company in 1996? Who was a chairman of Merck & Co. in 1996?
75.7	Name products manufactured by Merck. Name products manufactured by Merck & Co.

Table F.1: Examples of combining questions and their associated target.

result in a independent question, for example question 75.5 cannot be answered without reference to the target. In cases such as these the target is simply appended to the question to enable relevant documents to be located. It should be clear, however, that in the other questions the target has been successfully inserted into the question including the addition of possessives where necessary.

In this evaluation 40 of the 455 factoid and list questions could not be modified to insert the target. This has consequences during retrieval and answering of the question. The insertion failed mainly because the reference to the target was made by a nominal expression or an ellipsis instead of a pronominal expression (*"the first flight"* for space shuttles or *"the center"* for Berkman Center for Internet and Society). Of the remaining 405 questions whilst the target was inserted in the question in a few cases this resulted in badly formed or misleading questions. For example question 70.4 was *"What was the affiliation of the plane?"* for the target *"Plane clips cable wires in Italian resort"* for which this approach produced the question *"What was the affiliation of Plane clips cable wires in Italian resort?"*.

The following sections detail the results for the three different question types. It is interesting to note that all three results are above the median (and mean) of the submitted systems, which is a definite improvement over the prototype systems evaluated as part of TREC 2003 (see Appendix D) and continues the work evaluated as part of TREC 2004. Of the 30 participating groups the `shef05lmg` run was ranked 9^{th} (when all three question types were combined) with it ranking 20^{th} out of the 71 submitted runs.

Factoid Questions

The factoid component of the evaluation consisted of 362 questions. The official published statistics for the run were:

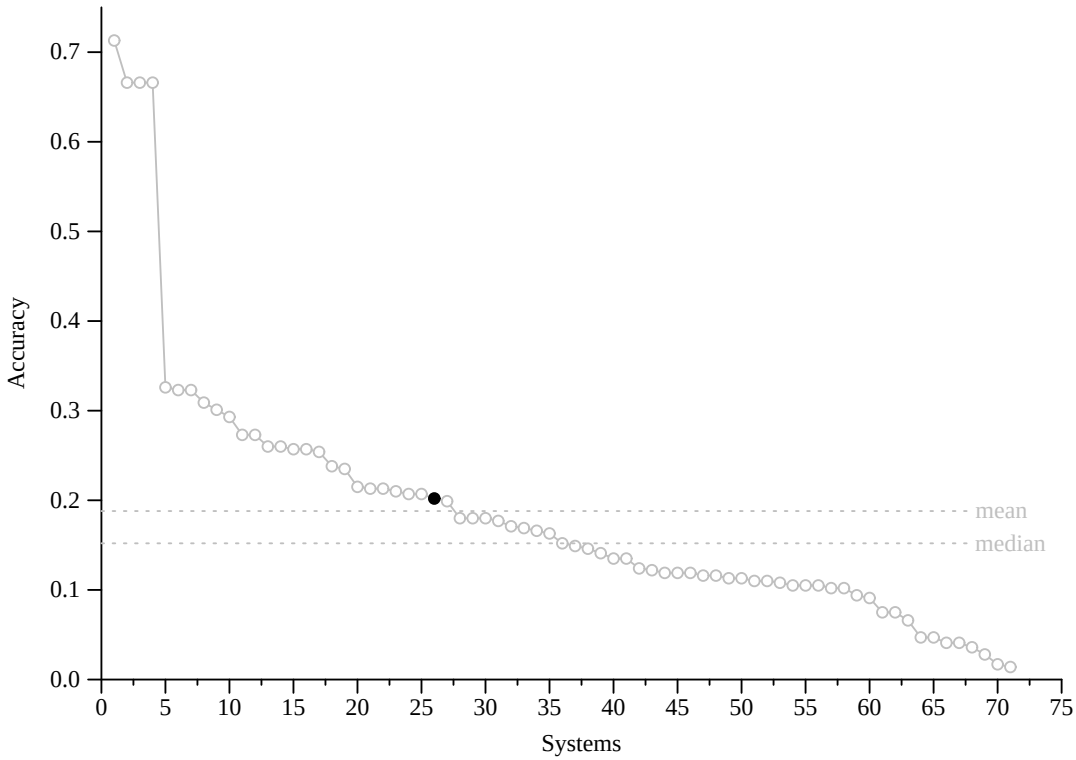

Figure F.1: Official TREC 2005 factoid scores (● is `shef05lmg`).

		Correct	Incorrect	Total
Question Analysis	Assigned UNKNOWN	32	31	63
	Assigned known type	259	40	299
	Total	291	71	362
Answer Extraction	Returned NIL	4	76	80
	Surface Matching Patterns	1	2	3
	Semantic Type Extraction	68	209	277
	WordNet Extension	0	2	2
	Total	73	289	362

Table F.2: Summary of TREC 2005 factoid performance.

Number wrong (W): 271
Number unsupported (U): 6
Number inexact (X): 12
Number right (R): 73
Accuracy: 0.202
Precision of recognizing no answer: 4 / 80 = 0.050
Recall of recognizing no answer: 4 / 17 = 0.235

Figure F.1 shows how these results compare with those of the other runs submitted as part of the evaluation.

Question analysis was such that of the 362 questions only 299 were assigned a type while 63 were labelled UNKNOWN. Of the 299 typed questions 291 were assigned the correct type and 31 of the ones labelled as UNKNOWN should have been assigned a type by the

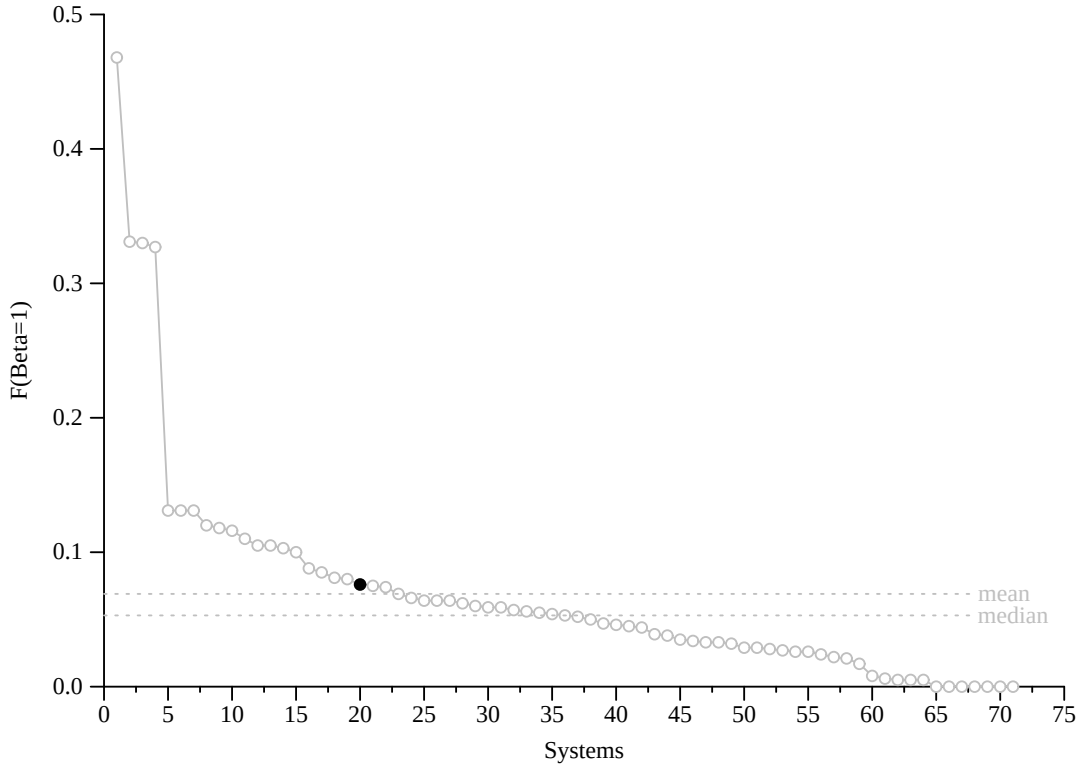

Figure F.2: Official TREC 2005 list scores (● is `shef05lmg`).

classifier. This gives a question classification accuracy of 80.4% – this means that the maximum attainable score after question classification is 0.804.

The distribution of questions between the different answer extraction components is shown in Table F.2. These results show that the majority of the correct answers were returned by the semantic type extraction approach of Section 8.1. This is mainly due to the lack of pattern sets developed to use as surface matching patterns rather than any problem with the approach. The official score of 0.202 is 25% of the maximum attainable score after question classification. This is a similar to the TREC 2004 performance reported in Appendix E, whilst the task has become harder with the addition of events as possible targets.

List Questions

The official F(β=1) score over the 93 list questions in the evaluation was 0.076. Figure F.2 shows how this result compares with the other submitted runs.

Of the 93 questions 20 could not be answered using the approaches to answering factoid questions detailed in this thesis. For these questions it was assumed that the answer would be one or more noun phrases, and so all noun phrases in the documents retrieved for the questions were extracted and then ranked in the same manner as answers found by the semantic type extraction system. This had the affect of slightly increasing the F measure

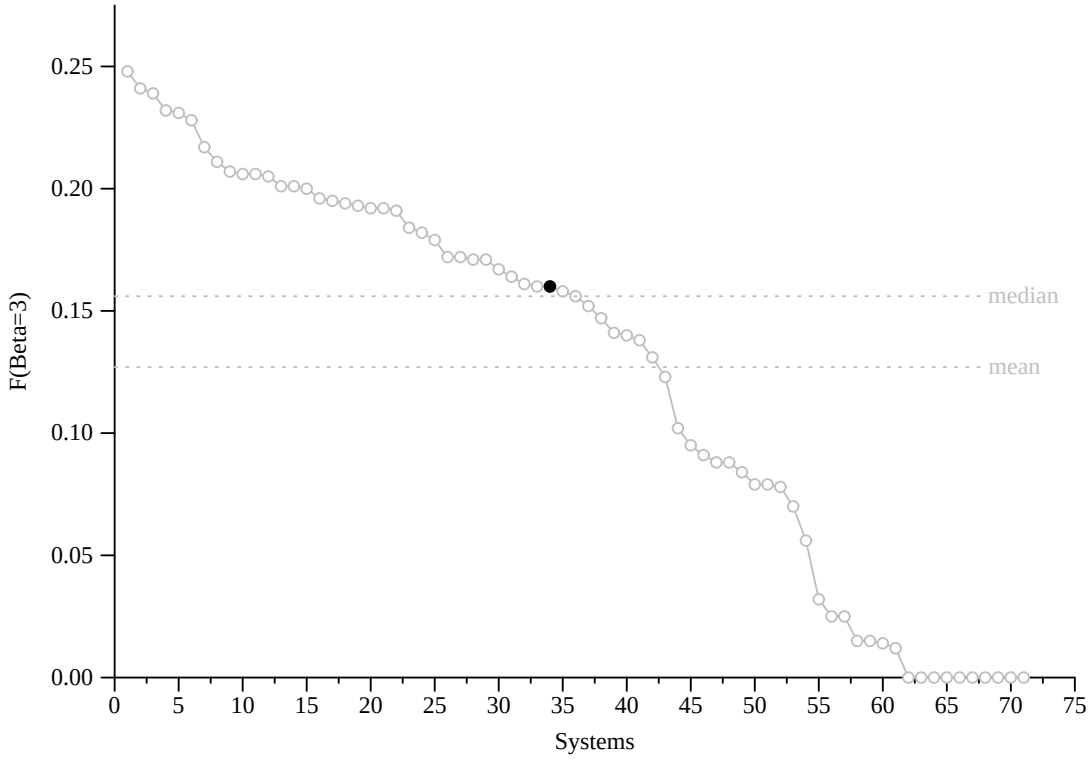

Figure F.3: Official TREC 2005 definition scores (● is `shef05lmg`).

score from 0.072 to the official score of 0.076 as an extra 5 distinct answers were found using this method.

The number of answers to return for each question, regardless of how they were extracted, was calculated in the same as in TREC 2004 (see Appendix E) which limits the number of returned answers in a bid to boost precision while not overly affecting recall so as to maximize the F(β=1) score for each question.

Definitions

The official average F(β=3) score for the 75 definition questions used in the evaluation was 0.160. Figure E.3 shows how this result compares with those of the other runs submitted for evaluation.

Of the 75 definition questions, the system created a definition for all but six questions. These failures to produce a definition were due to the target processing not being able to correctly determine the focus and qualification (see Section 11.1). Four of the six targets were organizations together with their common abbreviation (e.g. *"United Parcel Service (UPS)"*). For these targets the focus was wrongly determined to be just the first word which led to poor IR and the inability to find relevant sentences from which to construct the definition.

Of the 69 definitions which the system constructed only 38 of them contained at least one nugget deemed vital by the assessors. There were 15 definitions which did contain one or more non-vital nuggets but which were nevertheless awarded a score of 0 due to the evaluation metric (see Section 3.1.3). The fact that the evaluation metric insists that at least one vital nugget be present for the definition to achieve an non-zero score means that the performance of the systems is not truly represented. A possible approach to evaluating definition questions which does not suffer in the same way is described by Lin and Demner-Fushman (2005). Their approach applies weights to each nugget to represent their importance rather than making a binary decision between vital and non-vital.

It is interesting to note that of the 23 questions for which no nuggets were retrieved 9 were of the new target type event. In fact the definitions for 14 of the 18 event targets contained no vital nuggets and hence scored 0. This shows that the system developed in Part III of this thesis needs further development before it is capable of producing useful definitions of events.

Bibliography

Steven Abney, Michael Collins, and Amit Singhal. 2000. Answer Extraction. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP 2000)*, pages 296–301, Seattle, Washington, USA.

Eugene Agichtein, Steve Lawrence, and Luis Gravano. 2001. Learning Search Engine Specific Query Transformations for Question Answering. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*, pages 169–178, Hong Kong, May 1-5.

Kisuh Ahn, Johan Bos, Stephen Clark, James R. Curran, Tiphaine Dalmas, Jochen L. Leidner, Matthew B. Smillie, and Bonnie L. Webber. 2004. Question Answering with QED and Wee at TREC-2004. In *Proceedings of the 13th Text REtrieval Conference*.

Pranav Anand, Eric Breck, Brianne Brown, Marc Light, Gideon S. Mann, Ellen Riloff, Mats Rooth, and Michael Thelen. 2000. Fun with Reading Comprehension. Johns Hopkins Summer Workshop.
`http://www.clsp.jhu.edu/ws2000/groups/reading/index.shtml` [September 2002].

John W. Backus, Friedrich L. Bauer, Julien Green, C. Katz, John L. McCarthy, Alan J. Perlis, Heinz Rutishauser, Klaus Samelson, Bernard Vauquois, Joseph H. Wegstein, Adriaan van Wijngaarden, Michael Woodger, and Peter Nauer. 1963. Revised Report on the Algorithm Language ALGOL 60. *Communications of the ACM*, 6(1):1–17.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison Wesley.

Tim Berners-Lee. 1998. Semantic Web Road Map.
`http://www.w3.org/DesignIssues/Semantic` [November 2004].

Matthew W. Bilotti, Boris Katz, and Jimmy Lin. 2004. What Works Better for Question Answering: Stemming or Morphological Query Expansion. In *Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering (IR4QA)*, pages 1–7, Sheffield, UK, July 29.

Sasha Blair-Goldensohn, Kathleen R. McKeown, and Andrew Hazen Schlaikjer. 2003. A Hybrid Approach for Answering Definitional Questions. In *Proceedings of the 12th Text REtrieval Conference*.

Daniel G. Bobrow, Ronald M. Kaplan, Martin Kay, Donald A. Norman, Henry Thomson, and Terry Winograd. 1977. GUS, a Frame Driven Dialog System. *Artificial Intelligence*, 8(2):155–173.

Eric Breck, John D. Burger, Lisa Ferro, David House, Marc Light, and Inderjeet Mani. 1999. A Sys Called Qanda. In *Proceedings of the 8th Text REtrieval Conference*.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-Intensive Question Answering. In *Proceedings of the Tenth Text REtrieval Conference*.

Andrei Z. Broder. 1997. On the Resemblance and Containment of Documents. In *Proceedings of Compression and Complexity of SEQUENCES*, pages 21–29.

Sabine Bucholz and Walter Daelemans. 2001. Complex Answers: A Case Study using a WWW Question Answering System. *Natural Language Engineering*, 7(4):301–323.

Chris Buckley and Ellen M. Voorhees. 2004. Retrieval Evaluation with Incomplete Information. In *Proceedings of the 27th ACM International Conference on Research and Development in Information Retrieval (SIGIR 2004)*, pages 25–32, Sheffield, UK, July.

John D. Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George A. Miller, Dan Moldovan, Bill Ogden, John Prager, Ellen Riloff, Amit Singhal, Rohini Shrihari, Tomek Strzalkowski, Ellen M. Voorhees, and Ralph Weischedel. 2000. Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A). Technical report, NIST. `http://www-nlpir.nist.gov/projects/duc/roadmapping.html` [Septemper 2002].

Gavin Burnage, 1990. *CELEX - A Guide for Users*. Centre for Lexical Information, University of Nijmegen.

Nancy Chinchor. 1998. MUC-7 Named Entity Task Definition, Version 3.5. Fairfax, VA.

Cyril W. Cleverdon and Michael Keen. 1966. Factors Determining the Performance of Indexing Systems, Volume 2. Cranfield College of Aeronautics.

William W. Cohen. 1995. Fast Effective Rule Induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123.

John M. Conroy, Judith D. Schlesinger, John Goldstein, and Dianne P. O'Leary. 2004. Left-Brain/Right-Brain Multi-Document Summarization. In *Proceedings of the Document Understanding Conference (DUC 2004)*, Boston, MA, May 6-7.

Richard J. Cooper and Stefan M. Rüger. 2000. A Simple Question Answering System. In *Proceedings of the 9th Text REtrieval Conference*.

Jack Copeland. 1993. *Artificial Intelligence: A Philosophical Introduction*. Blackwell Publishers Ltd.

Ann Copestake and Karen Spärck Jones. 1990. Natural Language Interfaces to Databases. *The Knowledge Engineering Review*, 5(4):225–249.

Hang Cui, Min-Yen Kan, and Tat-Seng Chua. 2004a. Unsupervised Learning of Soft Patterns for Definitional Question Answering. In *Proceedings of the Thirteenth World Wide Web Conference (WWW 2004)*, pages 90–99, New York, May 17-22.

Hang Cui, Min-Yen Kan, Tat-Seng Chua, and Jing Xiao. 2004b. A Comparative Study on Sentence Retrieval for Definitional Question Answering. In *Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering (IR4QA)*, pages 9–16, Sheffield, UK, July 29.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, Philadelphia.

Tiphaine Dalmas and Bonnie L. Webber. 2004. Answer Comparison: Analysis of Relationships between Answers to 'Where'– Questions. In *Proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics (CLUK '04)*, Birmingham, UK, January 7.

Tiphaine Dalmas, Dan Shen, Sebastian Riedel, and Jochen L. Leidner. 2004. Towards an Automated Mixed-Media Answer Summary System: A Next-Generation Question Answering System . An IGK Summer School Project. `http://www.coli.uni-sb.de/egk/Meeting-04/Projects/media/` [November 2004].

V. L. Deglin and Marcel Kinsbourne. 1996. Divergent Thinking Styles of the Hemispheres: How Syllogisms are Solved During Transitory Hemisphere Suppression. *Brain and Cognition*, 31:285–307.

Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web Question Answering: Is More Always Better? In *Processdings of the 25th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pages 291–298, Tampere, Finland, August.

Daniel M. Dunlavy, John M. Conroy, Judith D. Schlesinger, Sarah A. Goodman, Mary Ellen Okurowski, Dianne P. O'Leary, and Hans van Halteren. 2003. Performance of a Three-Stage System for Multi-Document Summarization. In *Proceedings of the Document Understanding Conference (DUC 2003)*.

Abdessamad Echihabi, Ulf Hermjakob, and Eduard Hovy. 2003. Multiple-Engine Question Answering in TextMap. In *Proceedings of the 12th Text REtrieval Conference*.

David Eichmann and Padmini Srinivasan. 1999. Filters, Webs and Answers: The University of Iowa TREC-8 Results. In *Proceedings of the 8th Text REtrieval Conference*.

Joel L. Fagan. 1987. Automatic Phrase Indexing for Document Retrieval. In *Proceedings of the 10th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 91–101, New Orleans, Louisiana, United States.

Robert Gaizauskas, Mark A. Greenwood, Mark Hepple, Ian Roberts, Horacio Saggion, and Matthew Sargaison. 2003. The University of Sheffield's TREC 2003 Q&A Experiments. In *Proceedings of the 12th Text REtrieval Conference*.

Robert Gaizauskas, Mark A. Greenwood, Mark Hepple, Ian Roberts, and Horacio Saggion. 2004. The University of Sheffield's TREC 2004 Q&A Experiments. In *Proceedings of the 13th Text REtrieval Conference*.

Robert Gaizauskas, Mark A. Greenwood, Henk Harkema, Mark Hepple, Horacio Saggion, and Atheesh Sanka. 2005a. The University of Sheffield's TREC 2005 Q&A Experiments. In *Proceedings of the 14th Text REtrieval Conference*.

Robert Gaizauskas, Mark Hepple, Horacio Saggion, Mark A. Greenwood, and Kevin Humphreys. 2005b. SUPPLE: A Practical Parser for Natural Language Engineering Applications. Technical Report CS–05–08, Department of Computer Science, University of Sheffield.

Robert Gaizauskas, Mark Hepple, Horacio Saggion, Mark A. Greenwood, and Kevin Humphreys. 2005c. SUPPLE: A Practical Parser for Natural Language Engineering Applications. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT '05)*, Vancouver, Canada.

Bert F. Green, Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. 1961. BASEBALL: An Automatic Question Answerer. In *Proceedings of the Western Joint Computer Conference*, volume 19, pages 219–224. Reprinted in (Grosz et al., 1986), pages 545–549.

Mark A. Greenwood and Robert Gaizauskas. 2003. Using a Named Entity Tagger to Generalise Surface Matching Text Patterns for Question Answering. In *Proceedings of the Workshop on Natural Language Processing for Question Answering (EACL03)*, pages 29–34, Budapest, Hungary, April 14.

Mark A. Greenwood and Horacio Saggion. 2004. A Pattern Based Approach to Answering Factoid, List and Definition Questions. In *Proceedings of the 7th RIAO Conference (RIAO 2004)*, pages 232–243, Avignon, France, April 27.

Mark A. Greenwood, Ian Roberts, and Robert Gaizauskas. 2002. The University of Sheffield TREC 2002 Q&A System. In *Proceedings of the 11th Text REtrieval Conference*.

Mark A. Greenwood. 2001. Implementing a Vector Space Document Retrieval System. Department of Computer Science, The University of Sheffield, UK. `http://www.dcs.shef.ac.uk/~mark/phd/work/index.html` [October 2002].

Mark A. Greenwood. 2004a. AnswerFinder: Question Answering from your Desktop. In *Proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics (CLUK '04)*, pages 75–80, Birmingham, UK, January 7.

Mark A. Greenwood. 2004b. Using Pertainyms to Improve Passage Retrieval for Questions Requesting Information About a Location. In *Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering (IR4QA)*, pages 17–22, Sheffield, UK, July 29.

Grolier Electronic Publishing. 1990. The Academic American Encyclopedia.

Barbara J. Grosz, Karen Spärck Jones, and Bonnie L. Webber, editors. 1986. *Readings in Natural Language Processing*. Morgan Kaufmann.

Louise Guthrie, Roberto Basili, Fabio Zanzotto, Kalina Bontcheva, Hamish Cunningham, Marco Cammisa, Jerry Cheng-Chieh Liu, Jia Cui, Cassia Faria Martin, David Guthrie, Kristiyan Haralambiev, Martin Holub, Klaus Machery, and Frederick Jelinek. 2003. Semantic Analysis for Data Sparsity Compensation (SPARSE). Johns Hopkins Summer Workshop. `http://www.clsp.jhu.edu/ws2003/groups/sparse/` [February 2005].

Kadri Hacioglu and Wayne Ward. 2003. Question Classification with Support Vector Machines and Error Correcting Codes. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL '03)*, pages 28–30, Morristown, NJ, USA.

Sanda Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morărescu. 2000. FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of the 9th Text REtrieval Conference*.

Sanda Harabagiu, Dan Moldovan, Marius Paşca, Mihai Surdeanu, Rada Mihalcea, Roxana Gîrju, Vasile Rus, Finley Lăcătuşu, Paul Morărescu, and Răzvan Bunescu. 2001. Answering complex, list and context questions with LCC's Question-Answering Server. In *Proceedings of the 10th Text REtrieval Conference*.

John Haugland. 1985. *Artificial Intelligence: the Very Idea*. MIT Press.

J. F. Heiser, Kenneth M. Colby, William S. Faught, and Roger C. Parkison. 1980. Can Psychiatrists Distinguish a Computer Simulation of Paranoia from the Real Thing? *Journal of Psychiatric Research*, 15:149–192.

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

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep Read: A Reading Comprehension System. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 325–332, University of Maryland, USA, June 20-26.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2000. Question Answering in Webclopedia. In *Proceedings of the 9th Text REtrieval Conference*.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Towards Semantics-Based Answer Pinpointing. In *Proceedings of the DARPA Human Language Technology Conference (HLT)*, San Diego, CA.

Kevin Humphreys, Robert Gaizauskas, Mark Hepple, and Mark Sanderson. 1999. University of Sheffield TREC-8 Q & A System. In *Proceedings of the 8th Text REtrieval Conference*.

Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, Adwait Ratnaparkhi, and Richard J. Mammone. 2000. IBM's Statistical Question Answering System. In *Proceedings of the 9th Text REtrieval Conference*.

Betty R. Kirkwood. 1988. *Essentials of Medical Statistics*. Blackwell Science.

Julian Kupiec. 1993. MURAX: A Robust Linguistic Approach for Question Answering Using an On-Line Encyclopedia. In *Proceedings of the 16th Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 181–190, Pittsburgh, Pennsylvania, USA.

Wendy G. Lehnert. 1977. A Conceptual Theory of Question Answering. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 158–164, Cambridge, MA, USA, August. Reprinted in (Grosz et al., 1986), pages 651–657.

Xin Li and Dan Roth. 2002. Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, Taipei, Taiwan.

Marc Light, Gideon S. Mann, Ellen Riloff, and Eric Breck. 2001. Analysis for Elucidating Current Question Answering Technology. *Natural Language Engineering*, 7(4):325–342.

Jimmy Lin and Dina Demner-Fushman. 2005. Will Pyramids Built of Nuggets Topple Over? Technical Report LAMP-TR-127/CS-TR-4771/UMIACS-TR-2005-71, University of Maryland, College Park.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proceedings of the 2003 Language Technology Conferecne (HLT-NAACL 2003)*, Edmonton, Canada, May.

Jimmy Lin, Dennis Quan, Vineet Sinha, Karun Bakshi, David Huynh, Boris Katz, and David R. Karger. 2003. What Makes a Good Answer? The Role of Context in Question Answering. In *Proceedings of the Ninth IFIP TC13 International Conference on Human-Computer Interaction (INTERACT 2003)*, pages 25–32, Zurich, Switzerland, September.

Peter Lyman and Hal R. Varian. 2003. How Much Information? 2003. `http://www.sims.berkeley.edu/how-much-info-2003/` [February 2004].

Mark T. Maybury, editor. 2004. *New Directions in Question Answering*. MIT Press.

Diana Maynard, Kalina Bontcheva, and Hamish Cunningham. 2003. Towards a Semantic Extraction of Named Entities. In *Recent Advances in Natural Language Processing*, Bulgaria.

Chris McManus. 2002. *Right Hand, Left Hand: The Origins of Asymmetry in Brains, Bodies, Atoms and Cultures*. Orion Publishing Group.

George A. Miller. 1995. WordNet: A Lexical Database. *Communications of the ACM*, 38(11):39–41, November.

Tom M. Mitchell. 1997. *Machine Learning*. Computer Science Series. McGraw-Hill.

Ruslan Mitkov. 1995. Anaphora Resolution in Natural Language Processing and Machine Translation. Working paper, IAI, Saarbrücken.

Dan Moldovan, Sanda Harabagiu, Marius Paşca, Rada Mihalcea, Richard Goodrum, Roxana Gîrju, and Vasile Rus. 1999. LASSO - A Tool for Surfing the Answer Net. In *Proceedings of the 8th Text REtrieval Conference*.

Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morărescu, Finley Lăcătuşu, Adrian Novischi, Adriana Badulescu, and Orest Bolohan. 2002. LCC Tools for Question Answering. In *Proceedings of the 11th Text REtrieval Conference*.

Dan Moldovan, Sanda Harabagiu, Christine Clark, Mitchell Bowden, John Lehmann, and John Williams. 2004. Experiments and Analysis of LCC's two QA Systems over TREC 2004. In *Proceedings of the 13th Text REtrieval Conference*.

Diego Mollá Aliod, Jawad Berri, and Michael Hess. 1998. A Real World Implementation of Answer Extraction. In *Proceedings of the 9th International Conference on Database and Expert Systems Applications Workshop "Natural Language and Information Systems" (NLIS'98)*, pages 143–148, Vienna, August 26-28.

Christof Monz. 2003. *From Document Retrieval to Question Answering*. Ph.D. thesis, Institute for Logic, Language and Computation, University of Amsterdam. `http://www.illc.uva.nl/Publications/Dissertations/DS-2003-04.text.pdf` [April 2004].

Christof Monz. 2004. Minimal Span Weighting Retrieval for Question Answering. In *Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering (IR4QA)*, pages 23–30, Sheffield, UK, July 29.

Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Douglas Johnson. 2005. Terrier Information Retrieval Platform. In *Proceedings of the 27th European Conference on IR Research (ECIR 2005)*, volume 3408 of *Lecture Notes in Computer Science*, pages 517–519. Springer.

Marius Paşca. 2003. *Open-Domain Question Answering from Large Text Collections*. CSLI Studies in Computational Linguistics. CSLI Publications.

Judy Pearsall, editor. 2001. *The New Oxford Dictionary of English*. Oxford University Press.

Joseph Polifroni and Stephanie Seneff. 2000. Galaxy-II as an Architecture for Spoken Dialogue Evaluation. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000)*, Athens, Greece.

Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia.

Martin Porter. 1980. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137.

Paul Procter. 1978. *Longman Dictionary of Contemporary English*. Longman.

Purdue University Online Writing Lab. 2004. Conciseness: Methods of Eliminating Wordiness. `http://owl.english.purdue.edu/handouts/print/general/gl_concise.html` [July 2004].

Hong Qi, Jahna Otterbacher, Adam Winkel, and Dragomir R. Radev. 2002. The University of Michigan at TREC2002: Question Answering and Novelty Tracks. In *Proceedings of the 11th Text REtrieval Conference*.

Deepak Ravichandran and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 41–47, Pennsylvania.

Ellen Riloff and Michael Thelen. 2000. A Rule-Based Question Answering System for Reading Comprehension Tests. In *Proceedings of the ANLP/NAACL-2000 Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*, pages 13–19, Seattle, Washington, May 4.

Ian Roberts and Robert Gaizauskas. 2004. Evaluating Passage Retrieval Approaches for Question Answering. In *Proceedings of 26th European Conference on Information Retrieval (ECIR'04)*, pages 72–84, University of Sunderland, UK.

Stephen E. Robertson and Steve Walker. 1999. Okapi/Keenbow at TREC-8. In *Proceedings of the 8th Text REtrieval Conference*.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of the 3rd Text REtrieval Conference*.

Assaf Rozenblatt. 2003. New Question Answering Engine, `www.brainboost.com`. `http://www.searchguild.com/tpage727-0.html` [November 2004].

Horacio Saggion and Robert Gaizauskas. 2004. Mining On-Line Sources for Definition Knowledge. In *Proceedings of FLAIRS 2004*, Florida, USA.

Horacio Saggion, Robert Gaizauskas, Mark Hepple, Ian Roberts, and Mark A. Greenwood. 2004. Exploring the Performance of Boolean Retrieval Strategies for Open Domain Question Answering. In *Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering (IR4QA)*, pages 45–52, Sheffield, UK, July 29.

Gerard Salton and Chris Buckley. 1990. Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science*, 41(4):288–297.

Gerard Salton, Amit Singhal, Mandar Mitra, and Chris Buckley. 1997. Automatic Text Structuring and Summarization. *Information Processing & Management*, 33(2):193–207.

Matthew Sargaison. 2003. Named Entity Information Extraction and Semantic Indexing – Techniques for Improving Automated Question Answering Systems. MSc Dissertation, Department of Computer Science, The University of Sheffield, UK. `http://www.dcs.shef.ac.uk/teaching/eproj/msc2003/abs/m2ms.htm` [February 2003].

Roger C. Schank and Robert Abelson. 1977. *Scripts, Plans, Goals and Understanding*. Hillsdale.

Roger C. Schank. 1975. *Conceptual Information Processing*. Elsevier Science Inc., New York, NY, USA.

Sam Scott and Robert Gaizauskas. 2000. University of Sheffield TREC-9 Q & A System. In *Proceedings of the 9th Text REtrieval Conference*.

Stephanie Seneff, Ed Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue. 1998. Galaxy-II: A Reference Architecture for Conversational System Development. In *Proceedings of ICSLP 98*, pages 931–934, Sydney, Australia, November.

William M. Shaw, Judith B. Wood, Robert E. Wood, and Helen R. Tibbo. 1991. The Cystic Fibrosis Database: Content and Research Opportunities. *Library and Information Science Research*, 13(4):347–366.

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

Martin M. Soubbotin and Sergei M. Soubbotin. 2001. Patterns of Potential Answer Expressions as Clues to the Right Answers. In *Proceedings of the 10th Text REtrieval Conference*.

Mark Stevenson and Mark A. Greenwood. 2005. A Semantic Approach to IE Pattern Induction. In *Proceedings of the 43rd Annual Conference of the Association of Computational Linguistics (ACL 2005)*, Ann Arbor, Michigan, June 26-28.

Mark Stevenson. 2004. Information Extraction from Single and Multiple Sentences. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland, August.

Tomek Strzalkowski, Louise Guthrie, Jussi Karlgren, Jim Leistensnider, Fang Lin, Jose Perez-Carballo, Troy Strazheim, Jin Wang, and Jon Wilding. 1996. Natural Language Information Retrieval: TREC-5 Report. In *Proceedings of the 5th Text REtrieval Conference*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 224–231, Sapporo, Hokkaido, Japan.

Beth M. Sundheim, editor. 1995. *Proceedings of the Sixth Message Understanding Conference*, Columbia, MD, November. Morgan Kaufman.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering. In *Proceedings of the Twenty-Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–47, Toronto, Canada, July.

Alan M. Turing. 1950. Computing Machinery and Intelligence. *Mind*, 59:433–460.

Esko Ukkonen. 1995. On-line Construction of Suffix Trees. *Algorithmica*, 14(3):249–260.

Ellen M. Voorhees. 1993. Using WordNet to Disambiguate Word Senses for Text Retrieval. In *Processdings of the Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 171–180, Pittsburgh, PA.

Ellen M. Voorhees. 1999. The TREC 8 Question Answering Track Report. In *Proceedings of the 8th Text REtrieval Conference*.

Ellen M. Voorhees. 2001. Overview of the TREC 2001 Question Answering Track. In *Proceedings of the 10th Text REtrieval Conference*.

Ellen M. Voorhees. 2002. Overview of the TREC 2002 Question Answering Track. In *Proceedings of the 11th Text REtrieval Conference*.

Ellen M. Voorhees. 2003a. Evaluating Answers to Definition Questions. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, volume 2, pages 109–111.

Ellen M. Voorhees. 2003b. Overview of the TREC 2003 Question Answering Track. In *Proceedings of the 12th Text REtrieval Conference*.

Ellen M. Voorhees. 2004. Overview of the TREC 2004 Question Answering Track. In *Proceedings of the 13th Text REtrieval Conference*.

Joseph Weizenbaum. 1966. ELIZA - A Computer Program for the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 9:36–45.

Roy Williams. Mid-1990s. Data Powers of Ten.

Terry Winograd. 1972. *Understanding Natural Language*. Academic Press, New York.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. 1999a. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, second edition.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell, 1999b. *Managing Gigabytes: Compressing and Indexing Documents and Images*, chapter 4. In (Witten et al., 1999a), second edition.

William Woods. 1973. Progress in Natural Language Understanding - An Application to Lunar Geology. In *AFIPS Conference Proceedings*, volume 42, pages 441–450.

Jinxi Xu and W. Bruce Croft. 2000. Improving the Effectiveness of Information Retrieval with Local Context Analysis. *ACM Transactions on Information Systems*, 18(1):79–112.

Jinxi Xu, Ana Licuanan, and Ralph Weischedel. 2003. TREC2003 QA at BBN: Answering Definitional Questions. In *Proceedings of the 12th Text REtrieval Conference*.

Jinxi Xu, Ralph Weischedel, and Ana Licuanan. 2004. Evaluation of an Extraction-Based Approach to Answering Definitional Questions. In *Proceedings of the 27th ACM International Conference on Research and Development in Information Retrieval (SIGIR 2004)*, pages 418–424, Sheffield, UK, July.

Hui Yang, Hang Cui, Mstisalv Maslennikov, Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2003. QUALIFIER in TREC-12 QA Main Task. In *Proceedings of the 12th Text REtrieval Conference*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLP)*, pages 282–289, Seattle, WA.

Dell Zhang and Wee Sun Lee. 2003. Question Classification using Support Vector Machines. In *Proceedings of the 26th ACM International Conference on Research and Developement in Information Retrieval (SIGIR'03)*, pages 26–32, Toronto, Canada.

Zhiping Zheng. 2002. AnswerBus Question Answering System. In *Proceedings of the Human Language Technology Conference (HLT 2002)*, San Diego, CA, March 24-27.

Justin Zobel. 1998. How Reliable are the Results of Large-Scale Information Retrieval Experiments? In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–314, Melbourne, Australia, August.

Victor Zue, Stephanie Seneff, James Glass, Joseph Polifroni, Christine Pao, Timothy J. Hazen, and Lee Heatherington. 2000. JUPITER: A Telephone-Based Conversational Interface for Weather Information. *IEEE Transactions on Speech and Audio Processing*, 8(1):100–112.

Author Index

"If I have seen further it is by standing on the shoulders of giants."

Sir Isaac Newton, 5th February 1676