Question Answering

Mark Andrew Greenwood MEng
(Supervised by Dr R. Gaizauskas)

Department of Computer Science, University of Sheffield,
Regent Court, 211Portobello St, Sheffield S1 4DP, UK
mark@dcs.shef.ac.uk

August – September 2002

Abstract

This document reports the progress made during the first year of my studies into question answering – mainly concerned with work undertaken to allow us to take part in TREC 2002. The document also includes a comprehensive review of the history of question answering, covering early work in areas such as natural language interfaces to databases and reading comprehension systems. The report concludes with an outline of a proposal for the next two years of my studies – which aim to show the uses and benefits of natural language techniques to the field of question answering by examining them against the backdrop of a simply pattern matching system, similar in idea to those which have recently been shown to be highly successful.

Acknowledgements

The work detailed in this report would not have been possible without the support and encouragement of quite a few people who I would like to thank:

- My supervisor Dr R. Gaizauskas for the huge amount of time he has spent helping me familiarise myself with the research area, and for his numerous suggestions and comments.
- Professor M. Holcombe and Dr M. Hepple for their constructive comments and for asking those awkward questions designed to make me think harder.
- Ian Roberts for his help TREC and all the work he did on the information retrieval side of Question Answering.
- My parents for their continued support and encouragement and for having helped proof read this report.
- My girlfriend, Bryony Edwards, for her support and encouragement, and for having helped proof numerous drafts of this report.
- All the members of the NLP group who have answered questions and pointed me in the right direction.

Contents

Abstract i

Acknowledgements ii

Contents iii

1 Introduction 1

2 The Scope of Question Answering 2
  2.1 Applications and Their Users 2
  2.2 Questions 2
  2.3 Answers 4
  2.4 Evaluation 5
    2.4.1 Evaluation at TREC 5

3 A Brief History of Question Answering 8
  3.1 Natural Language Database Systems 8
  3.2 Dialog Systems 9
  3.3 Reading Comprehension Systems 13
  3.4 Open-Domain Questions and TREC 16
    3.4.1 The Document Collections 17
    3.4.2 Temporal and No Answer Questions 17
    3.4.3 Sub Tracks 18
    3.4.4 Performance of Systems at TREC 19
    3.4.5 The Future of TREC 21
  3.5 Other Techniques Relevant to Question Answering 21
    3.5.1 Inference Rules 21

4 Progress to Date 23
  4.1 Overview of the TREC 9 System 23
    4.1.1 Question Parsing 24
    4.1.2 Resolution of Question and Candidate Answer Texts 25
    4.1.3 Answer Output 26
  4.2 TREC 2001 26
  4.3 Changes and Updates to the QA System 27
    4.3.1 Processing Answer Instances 28
    4.3.2 Property Similarity 28
    4.3.3 Semantic Similarity 28
    4.3.4 Answer Scoring Algorithm 31
    4.3.5 Question - Answer Overlap 31
    4.3.6 Combining Semantically Similar Answers 31
    4.3.7 Strategy for Ranking Answers 32
  4.4 Boosting System Performance Using Answer Redundancy 33
  4.5 List Questions 34
  4.6 A Framework in which to Develop Grammars 35
  4.7 TREC 2002 36
    4.7.1 Broken Questions 36
    4.7.2 Other Bugs in the System 37

4.7.3 Answer Ranking ... 37
4.8 Question Answering over the World Wide Web ... 38
4.8.1 Comparison to Other WWW QA Systems ... 39

5 Issues and Challenges Facing Question Answering ... 41
5.1 How Important is Reliable Information Retrieval? ... 41
5.2 Are Natural Language Techniques Helpful? ... 44
5.3 Does Machine Learning Have a Role to Play? ... 45
5.4 Do Question Answering Systems Require World Knowledge ... 45

6 Future Work ... 48

7 Bibliography ... 51

Appendix 1: Index of Figures ... 54

Appendix 2: Index of Equations ... 55

Appendix 3: Index of Tables ... 56

1 Introduction

Ever since the dawn of spoken language humans have hungered for knowledge. We have explored the world around us by asking questions about what we can see and feel. As time progressed we became more and more interested in acquiring knowledge, constructing schools and universities to teach each new generation things their forefathers could never have imagined. With modern technology it is now easier to find any information than it has ever been in the history of human society.

With the recent explosive growth in the number of available electronic documents we are entering an age where effective question answering (QA) technology will become essential to being able to effectively access this vast collection of knowledge.

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

Unfortunately, the increase in the amount of electronic text available shows no sign of abating. Although modern search engines (such as Google[1]) are able to cope with the amount of text available, they are most useful when a user presents a query to the search engine which causes just a couple of documents to be returned, which the user can then manually search to find the relevant information.

It is becoming more and more the case, however, that a simple query using a modern search engine will return hundreds if not thousands of documents; more than can be easily 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 more direct access to this vast store of information.

Ideally a user may ask a question such as "*What is the state flower of Hawaii?*", instead of the user being presented with a list of documents, question answering technology would simply present the answer, "*Hibiscus*", and a link to the relevant document. This is the view of question answering that is currently prevalent due in no small part to the question answering track at the Text REtrieval Conferences (for detailed information about TREC see section 3.4).

[1] Google is a registered trademark of Google Inc.

2 The Scope of Question Answering

2.1 Applications and Their Users

Many different types of question answering systems have been envisaged all with a specific type of user in mind. One system of classifying the applications by the type of user was detailed in [Burg2000]. The authors defined a scale with four different user levels (and hence applications) as follows:

Level 1.	Casual Questioner
Level 2.	Template Questioner
Level 3.	Cub Reporter
Level 4.	Professional Information Analyst

On this scale the questions become more difficult and hence the applications become more complex, for example the two ends of the spectrum are defined in Table 2-1 with the other points on the scale falling between these two extremes.

	Casual Questioner	*Professional Information Analyst*
Questions	Simple facts	Complex, uses judgement terms, knowledge of user context needed, broad scope
Answers	Simple answers found in a single document	Search multiple sources (in multiple media/languages), fusion of information, resolution of conflicting data, multiple alternatives, adding interpretation, drawing conclusions

Table 2-1: Comparison of Casual Questioner and Professional Information Analyst.

Clearly this does not fully define the variation in the different possible applications and users. Other possibilities include domain-specific question answering (i.e. help systems), or even applying question answering to images, sounds or any other large collection of data which about which a user may wish to ask questions.

2.2 Questions

Clearly there are many different types of questions that people will want to ask. Researchers have often categorised questions based on the words they contain, i.e. Who, What, When, Where, etc. This categorization is linguistically useful, but it tells us nothing about how difficult the questions are for a system to answer, or even if they will be able to answer them at all!

Moldovan et al. (see [Mold2002]) define 5 question classes of increasing complexity and the system requirements necessary to answer them:

1. *QA systems capable of processing factual questions.* These systems extract answers from one or more documents. Often the answer is found verbatim in a text or as a simple morphological variation.
2. *QA systems enabling simple reasoning mechanisms.* The characteristic of this class is that answers are found in snippets of text, but unlike in 1, inference is necessary to relate the question with the answer. An example is *"How did Socrates die?"* where *die* has to be linked with *drinking poisoned wine*.

3. *QA systems capable of answer fusion from different documents.* In this class the partial answer information is scattered throughout several documents and answer fusion is necessary. The complexity of questions here ranges from assembling simple lists to far more complex script-like answers (e.g. *"How do I assemble a bike?"*).
4. *Interactive QA systems.* These systems are able to answer questions in the context of previous interactions with the user.
5. *QA systems capable of analogical reasoning.* The characteristic of these systems is their ability to answer speculative questions similar to: *"Is the airline industry in trouble?"*

Also contained in the paper by Moldovan et al. is Table 2-2 showing the distribution of the TREC questions (using all the QA questions from TRECs 8, 9 and 2001 including the list and context questions – see section 3.4 for details about the different TRECs) across these five different classes of question.

Type	*Number (%)*
Class 1 (factual)	985 (67.5%)
Class 2 (simple reasoning)	408 (27.9%)
Class 3 (fusion – list)	25 (1.7%)
Class 4 (interactive – context)	42 (2.9%)
Class 5 (speculative)	0 (0.0%)

Table 2-2: Class distribution of TREC questions.

The best scoring systems, entered into TREC, are able to answer approximately two thirds of the questions put to them. This matches almost exactly with the percentage of Class 1 questions given in the above table.

This type of categorization is much more useful than one based on the question words (who, what, etc.) and may prove to be a useful way of categorising a systems capability. For example a system may only be able to reliably answer Class 1 questions, but if the user is aware of this limitation then the questions they ask will be more likely to be correctly answered and hence they will be happier with the performance of the system. For example a reporter researching the background to a story will know only to ask questions that will have a factual answer. For instance if they were working on the Concorde crash which happened in 2000 then they may have asked questions such as:

"Has Concorde ever crashed before?"
"When was the last air crash in France?"

they would, however, know not to ask question such as:

"Will the authorities ground the remaining Concorde's while they investigate the crash?"

because this is a speculative (i.e. Class 5) question which they know the system will be unable to answer. In simple terms labelling a system based on the class of questions they can answer will limit user disappointment, as they will only expect the system to be able to answer certain types of question. For this to be useful, however, enough groups will have to adopt a single categorisation and apply it rigidly otherwise there will be no appreciable benefit.

2.3 Answers

A precise all encompassing definition of what encompasses an answer is a difficult think to come by. 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 many different ways. Most of the systems we will look at in the remainder of this paper use unstructured text as their source of answers, and usually (but not always) simply return, as an answer, a short extract from the text. The major question with this type of system is how long the returned answer should be. To this end TREC 2002 is insisting that the answers returned are *exact* although even that is difficult to define. The reason for insisting on an exact answer is to see how good systems are at pinpointing the exact answer. Most useable systems would want to return more than an exact answer, i.e. they would want to place the answer in the same context as it was found in the document or present some other form of justification. Isolating the exact answer does, however, have its benefits, as it allows varying length strings to be taken from the document correctly centred on the answer, and also it opens up the possibility of using the exact answer along with text generation systems to provide answers which are not simply cut from the text.

The main problem with asking systems to return exact answers is the definition of what makes an answer exact. Quite a long discussion on this took place on the mailing list for participants in TREC 2002 and the following sums up the feelings of most of the participants:

> It was suggested by L. Plamondon that an exact answers is usually a noun phrases. This would mean that questions such as "*Which river is known as the 'Big Muddy'?*" can have all of the following as valid exact answers: *Mississippi*, *the Mississippi*, *Mississippi river* and *the Mississippi river*. L. Liddy pointed out, however, that certain types of 'what' questions can have a verb phrase as an exact answer. Most interesting were the comments made by J. Prager: "…it is important to distinguish extraneous material that is junk from extra material that further answers the question. I just did a search with *'What does a red wolf weigh?'* and got a document that included the text *'females weighing 40-60 pounds and males weighing 60-80 pounds'*. Isn't this text a much better answer than just, say, *'60 pounds'*? In other words, isn't that the kind of answer we should be trying to get our systems to give?"

Clearly it is very difficult for people to make a definition of what constitutes an exact answer, which will satisfy every research group. Once the results of TREC 2002 are released (around the 1st of October 2002) this discussion will surely resurface, as some groups will undoubtedly be unhappy about some of the answers deemed to be non-exact. My main concern lies with questions whose answers are date related. For example, will a question such as *"Which year…"* expect only a year to be returned as an exact answer or will a full date also be accepted, for example will the question *"What year did the shuttle Challenger explode?"* have only *1986* as an exact answer, or will *January 1986* be excepted. Note that *January 1986* is a more exact answer than *1986* to the question *"When did the shuttle Challenger explode?"*.

2.4 Evaluation

Evaluation can be subjective, especially when dealing with certain types of natural language systems. It is easy to evaluate systems in which there is a clearly defined answer (i.e. named entity recognition), 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. This is no easy task even for collections as small as the Cystic Fibrosis Database (see [Shaw1991]), which contains 1239 articles and is approximately 5 megabytes in size[2]. Imagine trying to do the same for the collection used for TREC 2001[3], which contained approximately 979,000 articles in 3033 megabytes of text [Voorh2001].

The LUNAR system was designed to answer questions about the geology of moon rocks and is of interest to this section as it was one of the first question answering systems to be subject to user-evaluation (see section 3.1 for more details on LUNAR and how it fared). More recently evaluation of QA systems has focused mainly on the QA track at the Text REtrieval Conferences (TREC) organised by the National Institute for Science and Technology (NIST).

2.4.1 Evaluation at TREC

For the first two QA tracks held within the TREC competition (TRECs 8 and 9) each system was allowed to return an ordered list of five possible answers to each question. Although the systems were returning multiple answers for each question there was no attempt to evaluate the confidence a system had in the answers other than the order in which they were returned. At TREC 2001 the issue of confidence was addressed (in part) by requesting that for each question the system should state which of the five answers they were most confident in or state *UNSURE* to indicate that the system was not confident in any of the proposed answers.

Unfortunately this attempt at introducing confidence to the track was not especially successful as most systems always stated that they were confident in their first answer and hence the percentage of questions for which the final answer was correct and the system was sure was quite low (on average only 18.37%)

In this years track (TREC 2002) the idea of confidence has been taken a step further and tightly integrated into the evaluation. Now each system can only return one *exact answer* (see section 2.3 for a discussion of what constitutes an exact answer) for each question but more importantly the answers for the different questions must be presented in the submission file with the answer in which the system has most confidence appearing first. The scoring algorithm then rewards systems which are confident in their correct answers. The results for TREC 2002 are not yet available so we do not have any idea as to how good systems are at assigning confidence measures to their answers.

[2] The collection is available from http://www.dcc.ufmg.br/irbook/cfc.html in both ASCII text and in the form of XML documents.
[3] Note that TREC 2001 is also referred to as TREC 10 and TREC 2002 as TREC 11 depending on which papers you read.

For the first three QA tracks at TREC (TRECs 8, 9 and 2001) the score for a single question was the reciprocal rank of the first correct answer (or zero if no correct answer was given). The overall system score was then the mean of the scores for all the questions asked, known as Mean Reciprocal Rank (MRR), which is defined as:

$$MRR = \frac{\sum_{i=1}^{q} 1/r_i}{q}$$

Equation 2-1: Equation for calculating MRR.

In which q is the number of questions and r_i is the rank of the first correct answer for question i (or 0 if no correct answer is returned).

In the first year (TREC 8) all the questions were generated by NIST and were usually back formulation of statements in a document, i.e. for the text *"Mozart was born in 1756"* the likely question would be *"When was Mozart born?"*. In other words the questions were relatively simple and most (if not all) would have fallen into the class of factual questions (Class 1) as described in section 2.2.

During these first three question answering tracks participants submitted a single file to NIST for evaluation. An excerpt from a submission is shown below:

```
1008 Q0 AP880531-0196  1 0.723 shef Hibiscus
1008 Q0 AP880531-0196  2 0.522 shef GEORGIA - Tiger swallowtail
1008 Q0 WSJ890815-0098 3 0.589 shef Connecticut's  8,800
1008 Q0 AP880324-0081  4 0.522 shef all tax food
1008 Q0 WSJ890815-0098 5 0.589 shef AIDS
1009 Q0 WSJ920116-0078 1 0.357 shef Mr. Wurzelbacher
1009 Q0 WSJ920212-0064 2 0.536 shef United Team
1009 Q0 AP900521-0190  3 0.522 shef 1988
1009 Q0 AP890830-0144  4 0.589 shef Gerald Richman
1009 Q0 WSJ920211-0060 5 0.615 shef Soviet Union
```

The format of the file is as follows[4]:

- The first column is the question number.
- The second column is currently unused and should always be `Q0`.
- The third column is either the official document identifier of the document that justifies the answer OR the string `NIL`. If no answer is found.
- The fourth column is the rank of the answer, and the fifth column shows the score (integer or floating point) that generated the ranking.
- The sixth column is called the "run tag" and should be a unique identifier for the group and the method used.
- The last column is the answer-string. If the third column was `NIL`, this column should be empty.

Clearly lots of useful information that groups could provide about how they answered a question are not included as part of the evaluation procedure, just the textual answer is used to evaluate and rank the systems.

[4] Summarised from the TREC 2001 question answering guidelines.

Starting with TREC 2002 the measure used will be analogous to document retrieval's un-interpolated average precision and is defined to be:

$$\frac{\sum_{i=1}^{q} c_i / i}{q}$$

Equation 2-2: The TREC 2002 scoring measure.

In which q is the number of questions, i is the question number and c_i is the number of correct responses up to question i. This measure rewards systems that rank questions they answered correctly above those that they did not.

Due to the different evaluation method (mainly because only one answer is being returned per question) the format of the submission file is quite a bit different to that used in previous years. An excerpt from a submission file for TREC 2002 is shown below:

```
1494 shef NYT19981203.0051 Kipling
1494 shef 0.8839285714285714 100.0 1 2.0 true
1834 shef NYT19990402.0243 Judas
1834 shef 0.8773809523809524 100.0 126 1.0 true
```

The format of the file is as follows[5]:

The first line contains the response to be scored, in the format:

```
qid run-tag docid answer-string
```

where

- qid is the question number,
- run-tag is a unique identifier of the group and method used.
- docid is the id of the supporting document or the string `NIL` if no answer is in the collection.
- answer-string is the exact answer or empty if docid is `NIL`.

The second line for a question contains the justification. The format for this line is

```
qid run-tag justification
```

where qid and run-tag are the same as for line 1 and justification is a string of at most 1024 bytes.

Clearly this new submission format allows systems to include their justification for returning the answer they give. Some systems will probably return 1024 bytes of the sentence containing the answer. Some groups have suggested they may include their inference chains (although how much use these will be to any other research group is debatable). Our system simply returns the information it uses to rank the answers (see section 4.3.7 for details of what these attributes are). Until the results of TREC 2002 are available it is unclear how useful this justification will prove to be. Depending on how people have used the line in this year's evaluation tighter restrictions on what it can contain may be introduced in future years.

[5] Summarised from the TREC 2002 question answering guidelines.

3 A Brief History of Question Answering

It would be wrong to claim that interest in QA technology is a recent development in Natural Language Processing (NLP). In fact, one of the earliest papers on the subject *"Answering English Questions by Computer"* which was written by Simmons in 1965 [Simm1965], begins with the statement that the paper reviews no less than fifteen English language QA systems built over the previous five years. Clearly QA research is not a new area, although over the intervening time its aims have shifted slightly.[6]

3.1 Natural Language Database Systems

Two of the best-known early QA systems were BASEBALL and LUNAR (see [Green1961] and [Woods1973] respectively). The BASEBALL system was designed to answer questions about (funnily enough) baseball games which had been played in the American league over 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" [Woods1973]. 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, 12% failed for clerical reasons and 10% had more serious errors.

Although many of these early systems were sophisticated, even by modern standards, they were nearly all restricted to a limited 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. An example of this (taken from a modern natural language database interface[7]) would be the question *"List the authors who have written books about business"* for which an SQL (Structured Query Language) query such as the following would be generated:

```
SELECT firstname, lastname FROM authors, titleauthor, titles
WHERE authors.au_id = titleauthor.au_id
AND titleauthor.title_id = titles.title_id
```

In simple terms, these early systems usually 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 QA research today.

Most QA research that took place during the 1970's was in a similar vein to the systems already mentioned. More examples of early QA research along the lines of LUNAR and BASEBALL can be found in [Grosz1986]. For a comprehensive review of this type of natural language front-end work right through to 1990 see [Cope1990].

[6] Note that some of the history of QA, as presented here, is based on the work of Hirschman and Gaizauskas presented in [Hirs2001].
[7] The example system is Microsoft's English Query 2000, which is part of Microsoft SQL Server. See http://www.microsoft.com/sql/ for more information.

3.2 Dialog Systems

In his seminal 1950 Mind article "Computing Machinery and Intelligence", Alan Turing described an experiment aimed at settling the question of whether or not a computer is a capable of thought [Turing1950]. The Turing Test (as the experiment has become known) gave birth to research into computer systems capable of holding a meaningful conversation. A conversation (usually referred to as a dialog) by its very nature usually contains at least one question and therefore is relevant to this history of QA technology.

One of the earliest, and best known, of these Artificial Intelligence (AI) dialog systems was Weizenbaum's 1966 creation ELIZA [Weiz1966]. 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.).

The following is a typical conversation with Eliza[8]:

Men are all alike.
IN WHAT WAY?
They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE?
Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE
He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED
It's true. I am unhappy
DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
I need some help, that much seems certain.
WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
Perhaps I could learn to get along with my mother.
TELL ME MORE ABOUT YOUR FAMILY
My mother takes care of me.
WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU
My father.
YOUR FATHER
You are like my father in some ways.
WHAT RESEMBLANCE DO YOU SEE
You are not very aggressive but I think you don't want me to notice that.
WHAT MAKES YOU THINK I AM NOT VERY AGGRESSIVE
You don't argue with me.
WHY DO YOU THINK I DON'T ARGUE WITH YOU
You are afraid of me.
DOES IT PLEASE YOU TO BELIEVE I AM AFRAID OF YOU
My father is afraid of everybody.
WHAT ELSE COMES TO YOUR MIND WHEN YOU THINK OF YOUR FATHER
Bullies.
DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE

[8] Note that in all transcripts of machine-human conversations, presented in this paper, the human's speech is in *italics* and the computers output is in SMALL CAPITALS.

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?

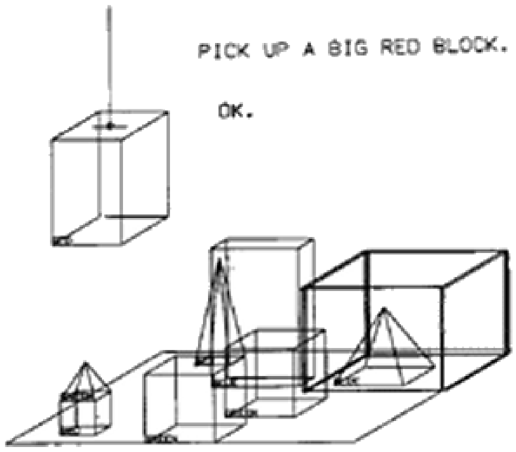
Clearly this is not a robust dialog system and there are many examples of ELIZA producing complete gibberish due to stringently applying these transformation rules. It should also be clear that this type of system is never going to be able to answer open-domain questions which require a real-world response, such as *"What do penguins eat?"*. For a complete exposé of ELIZA and a similar program called PARRY, developed by [Heiser1980], see [Cope1993].

Moving forward through the history of AI dialog systems research, we meet SHRDLU and GUS (see [Wino1972] and [Bobr1977] respectively). 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 3.1) are their dialog 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 and controlled a robot arm in a virtual micro-world which consisted of a table top strewn with coloured blocks of varying shapes and sizes and a box into which the blocks could be placed[9]. The following is a shortened version of the dialog that was used as a demo of SHRDLU:

Pick up a big red block.
OK.
Grasp the pyramid.
I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN.
Find a block which is taller than the one you are holding and put it into the box.
BY "IT", I ASSUME YOU MEAN THE BLOCK WHICH IS TALLER THAN THE ONE I AM HOLDING.
OK.
What does the box contain?
THE BLUE PYRAMID AND THE BLUE BLOCK.
What is the pyramid supported by?
THE BOX.
How many blocks are in the box?
FOUR OF THEM.
Can the table pick up blocks?
NO.
Can a pyramid be supported by a block?
YES.

[9] The original SHRDLU source code along with modern implementations can be found on the World Wide Web at http://hci.stanford.edu/cs147/examples/shrdlu/.

Can a pyramid support a pyramid?
I DON'T KNOW.
Stack up two pyramids.
I CAN'T.
Will you please stack up both of the red blocks and either a green cube or a pyramid?
OK.
Which cube is sitting on the table?
THE LARGE GREEN ONE WHICH SUPPORTS THE RED PYRAMID.
Is there a large block behind the pyramid?
YES, THREE OF THEM: A LARGE RED ONE, A LARGE GREEN CUBE, AND THE BLUE ONE.
Put a small one onto the green cube which supports a pyramid.
OK.
Put the littlest pyramid on top of it.
OK.
Does the shortest thing the tallest pyramid's support supports support anything green?
YES, THE GREEN PYRAMID.
What color is it?
BY "IT", I ASSUME YOU MEAN THE SHORTEST THING THE TALLEST PYRAMID'S SUPPORT SUPPORTS.
RED
How many things are on top of the green cubes?
I'M NOT SURE WHAT YOU MEAN BY "ON TOP OF" IN THE PHRASE "ON TOP OF GREEN CUBES".
DO YOU MEAN:
1 - DIRECTLY ON THE SURFACE
2 - ANYWHERE ON TOP OF?
2
THREE OF THEM.
Thank you.
YOU'RE WELCOME!


PICK UP A BIG RED BLOCK.

OK.

Figure 3-1: Original screen display of SHRDLU in action[10].

[10] Image used with the kind permission of Terry Winograd

Most people are bowled over by their first encounter with SHRDLU, however, the system is still severely limited to only discussing the micro-world it inhabits. Take the following exchange (this is a slightly modified version of a conversation, due to Haugeland [Haug1985], which appeared in [Cope1993]):

> *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 in fact 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"*.

Dialog 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 dialog systems. One modern dialog system is Jupiter (see [Zue2000]). The best description is taken from its product page at MIT[11]:

> *"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 (see [Sene1998] and [Poli2000] for details on GALAXY) and consists of the following stages:

1. *Speech Recognition*: converts the spoken sentence into text.
2. *Language Understanding*: parses the text into semantic frame – a grammatical structure containing the basic terms need to query the Jupiter database.

[11] The product page for Jupiter is at http://www/sls.lcs.mit.edu/sls/whatwedo/applications/jupiter.html.

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 dialog 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.

3.3 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 [Lehn1977]) researchers recognised this as an appropriate way of testing the language understanding abilities of computers and started to build systems to answer reading comprehension tests.

In a similar way to most of the early dialog 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 which was developed at the Yale AI lab. Consider the 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.*

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 (this transcription of the test comes from [Schank1977]):

> *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 HAD 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 especially interest is 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 the 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 he 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 can not 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 dialog 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 backward compared to systems such as Sam this limitation is partly based around 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 and Deep Read (see [Rilo2000] and [Hirs1999] respectively) both of which report results at between 30% and 40% in reading comprehension tests for children in the 3rd to 6th grades[12], an example being test concerning maple syrup in Figure 3-2.

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 NL techniques: Part of Speech (POS) tagging, stemming, name identification, semantic class identification and pronoun resolution.

[12] For those not familiar with the American grade schools (including the author), 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 3-2: An example reading comprehension test.

At first glance these systems seem exceptionally poor when compared with other QA systems, such as those entered in TREC, which at best answer approximately 70% of the questions. As was pointed out in [Anand2000], however, reading comprehension tests are 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".*

The benefit of multiple instances is discussed, in some detail, in relation to our QA system in section 4.4.

One modern system that attempts to return an actual answer rather than the sentence most likely to contain the answer is Spot (currently on version 5) which was developed at the Johns-Hopkins summer workshop in 2000 and is detailed in [Anand2000]. This work is based on the hypotheses that:

> *"... once 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), although if the system is evaluated between the final two stages then the performance is comparable with Quarc and Deep Read.

3.4 Open-Domain Questions and TREC

In recent years research in open-domain QA has been accelerated due to the inclusion of a QA track at the annual Text REtrieval Conference (TREC). This track was first run in 1999 with seventeen research groups entering one or more systems. Although the performance of the systems varied wildly, some of were remarkably good (see [Voorh1999] for an overview of the track and [Mold1999] for a report on the best overall system). An additional aim of the track was to define a task that would appeal to both the document retrieval community (as you could originally 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 QA track is now entering its fourth year, and contains not only the main QA track but also a list track in which systems are required to name a set number of entities, that meet some condition, with questions such as *"Name 20 countries that produce coffee"*.

The majority of the systems work in a similar fashion and consist of two main (often separate) sub-systems. Firstly an information retrieval (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 section 5.1.

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 technique, which rewards desirable words in the window. They moved the 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, among other options, tuning the window size and the score assigned to different words. As reported in [Hovy2001], although this form of answer pinpointing works to some degree (giving results of up to 30% in independent evaluations), this method has some serious limiting factors:

- 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 a semantic method for pinpointing answers.

3.4.1 The Document Collections

Two different document collections have been used over the four years the TREC QA track has now been running. For the first three years the collection was made up of a subset of the TIPSTER collection (which consists of 5 CDs of compressed text). By the end of the third year the set of documents being used for the QA task consisted of approximately 979,000 articles from numerous different news services including the Wall Street Journal and the Financial Times (see section 3.4.2 for a rundown of the exact contents).

As of TREC 2002 the document set has been changed to use the same collection as participants in ARDA's new AQUAINT program. This consists of documents from the following sources:

- AP newswire, 1998-2000
- New York Times newswire, 1998-2000
- Xinhua News Agency, 1996-2000

This new collection consists (according to NIST) of 1,033,461 documents. One point of concern with this new collection is some of the articles provided by the Xinhua News Agency. These articles appear to contain many spelling and grammar errors which may have an impact on the performance of systems (see section 5.4 for a discussion of world knowledge and document errors). Exactly what effect these errors will have on the performance of systems is unclear, although it may well make the evaluation more difficult (the documents are taken to be correct even if misspelled or misinformed), a clearer understanding of this problem may emerge once the results of the TREC 2002 evaluation have been released to the participants.

3.4.2 Temporal and No Answer Questions

In the first two TREC competitions each question was guaranteed to have a corresponding answer within the text collection. This constraint was dropped for TREC 2001 with 49 questions having no known answer. Systems were then able to return NIL as the supporting document if they believed there to be no answer.

Detecting whether or not a question has an answer is feasible, with one system having an accuracy of 0.76, but it is not trivial with only five of the sixty six runs having an accuracy of over 0.25 (where accuracy is the number of questions for which NIL was correctly returned divided by the total number of questions for which NIL was returned). As systems could return up to five ranked answers per question, some choose to always return NIL at rank five which resulted in an accuracy of only 0.1, but slightly increased their overall MRR score.

Our system makes no attempt to recognise questions that have no answer in the collection. Instead the system works by adding an initial NIL answer, with a zero score, to the ranked list of possible answers. All the answers found in the documents are then added to the list and as their scores will be greater than zero the NIL answer will be placed at the bottom of the ranked list. In other words the only time we ever return a NIL answer to a question is if we do not find any other possible answers within the documents.

There is a possible issue with the TREC competitions and the use of external resources. Many groups (including ours, see section 4.4, and Brill et al, see section 4.4

and [Brill2001]) make use of the web as a source of answers that are then matched against the closed TREC collection. This is inherently dangerous as a question, which has no relevant answer within the TREC collection will quite possibly have an answer on the WWW. If this answer is found and then projected onto the TREC collection there is always a chance that there will be a document which is relevant to both the query and answer without being a justification for that answer. One class of questions that can be a problem are those dealing with time.

The document collection used for the TREC 2001 conference, consists of approximately 979,000 documents from the following sources:

- AP newswire, 1988-1990
- Wall Street Journal, 1987-1992
- San Jose Mercury News, 1991
- Financial Times, 1991-1994
- Los Angeles Times, 1989-1990
- Foreign Broadcast Information Service, 1996

As can clearly be seen from the list of sources the latest events the news articles could cover would be the end of 1996, so a question such as *"When did the Kursk sink?"* which is answered by *"13th August 2000"* could not possibly be answered by any document in the collection. A system which utilises the Internet could, however, generate a correct answer but then either be unable to find a corresponding answer in the TREC collection or if like Brill et al. they simply look for a relevant document using an IR engine (especially if using an IR engine based on the vector space model) then they may find a relevant document but it certainly will not justify the answer found using the WWW.

A related problem is how to generate answers that are guaranteed not to have an answer in a closed collection, such as that used by TREC. Currently a question is suggested and then a brief search is made of the collection for an answer. If none can be found then it is assumed that it has no correct answer unless, during evaluation, a system proposes a correct answer that is backed up by a supporting document from within the collection. So even if the intention is to include say ten no-answer questions by the end of the evaluation there may only be five or so no-answer questions remaining.

The only sure way of introducing questions that do not have an answer in a closed collection is to use temporal question such as that mentioned above. Asking *"When did the Kursk sink?"* is a safe no-answer question if you know the contents of the collection stop at 1996 and the event you are asking about took place in 2000.

3.4.3 Sub Tracks

At TREC 2001 two new sub tracks were added to the main question answering track. These were concerned with answering list and context questions.

The list questions were designed to test the ability of systems to construct an answer to a question from multiple documents by requesting a set number of entities i.e. *"Name 20 countries that produce coffee"*. The collection was guaranteed to contain at least the requested number of entities but it was also guaranteed that more than one document would be needed to answer the question. As with the main track, the abilities of the competing systems varied wildly from the system entered by the Language Computer Corporation (see [Hara2001]), which achieved an average precision of 0.76, to the systems entered by Korea Advanced Institute of Science and Technology (see [Oh2001]) and Université de Montréal (see [Plam2001]) whose systems both achieved an average precision of only 0.07. This variation in ability of the systems is similar to that shown in the main track where the lowest MRR was 0.003 and the highest was 0.676.

The context questions are slightly different to the normal type of question asked. The aim of this sub-track is to test the ability of systems to track discourse objects across multiple questions (i.e. to have some idea of the interaction which has already taken place with the user). One of the question sets used in this evaluation was the following:

1. Which museum in Florence was damaged by a major bomb explosion in 1993?
2. On what day did this happen?
3. Which galleries were involved?
4. How many people were killed?
5. Where were these people located?
6. How much explosive was used?

Clearly the ability to answer the later questions intelligently depends on the systems ability to resolve referential links across the questions.

In total seven runs were submitted to this task, unfortunately the results were not quite what were expected. The ability of systems to answer questions later in a series did not correlate with their ability to answer the earlier questions. Rather the early questions allowed system to produce a document set (from the main TREC collection) that the later questions could then be answered against. Hence the ability to answer any question in the series was simply the ability to answer that particular type of question (i.e. who, what …).

3.4.4 Performance of Systems at TREC

As was mentioned in section 2.4.1, the first QA track consisted wholly of questions created by NIST as back formulations of text snippets. This lead to most of the questions being relatively simple to answer, with questions such as *"Who was the first American in space?"* having answers such as *"Alan Shepard was the first American in space"* available directly in the text collection. It was, however, the first time that QA systems had been evaluated in this way and so the task was harder than it might have been as research groups were unsure exactly what to expect and had little training data available.

TRECs 9 and 2001 used real questions taken from the logs of sites such as Ask Jeeves and MSN. This meant that the task was no harder, not only because the questions were not motivated by the text collection but also because there was now a significant number of definition type questions such as *"What is an eclipse?"*.

TREC 2001 was by far the most difficult of the three completed conferences as not only where the questions not based on the text but also not all of the questions were guaranteed to have an answer in the text collection.

Clearly the main task has been becoming more difficult year-on-year, although the performance of the systems has been maintained at a reasonable level, probably due in part to the experience gained from having competed in the previous years. This can be seen in Table 3-1 and Figure 3-3, which show the performance of the best, worst and mean systems as well as the best and worst performance of the Sheffield system at the three conferences[13].

	System MRR Score for				
TREC	Best	Worst	Mean	Sheffield's Best	Sheffield's Worst
8	0.660	0.002	0.237	0.081	0.071
9	0.580	0.038	0.218	0.206	0.159
10	0.676	0.003	0.236	0.343	0.169

Table 3-1: MRR scores of systems over TRECs 8, 9 and 10.

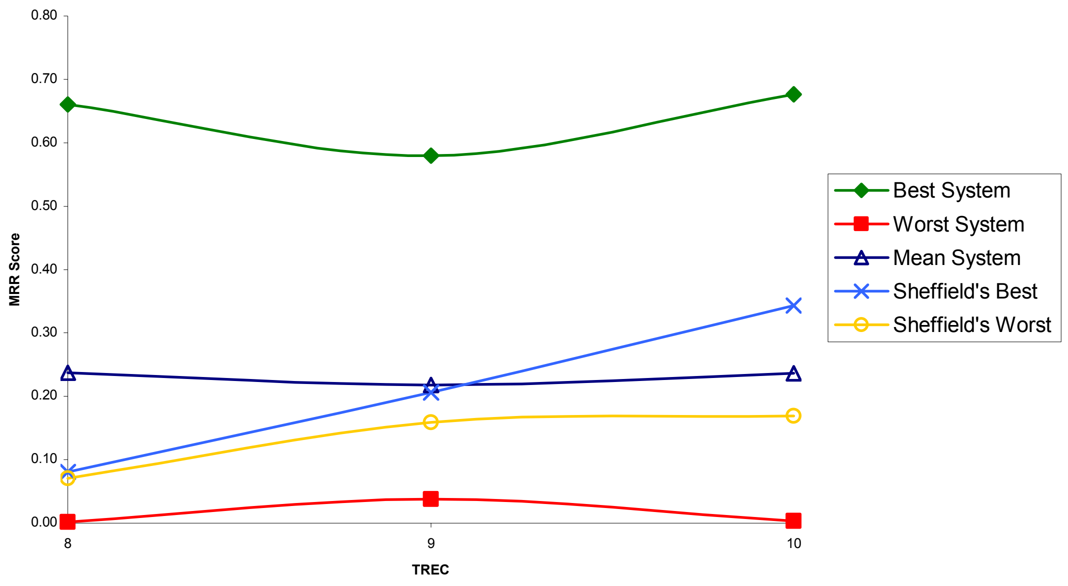


Figure 3-3: Performance of systems over TRECs 8, 9 and 10.

After TREC 2001 it was decided that in future the number of definition style questions will be monitored by NIST. This is partly because they are difficult to answer and to judge but also as there are more efficient ways of answering these type of questions other than using news wire texts (i.e. these questions could probably be better answered by having a QA system as an interface to an encyclopaedia).

[13] The results for the Sheffield system at TREC 10 are not official results, as the group did not enter a system. Rather these show the worst and best performance we have obtained from running and judging the questions ourselves.

3.4.5 The Future of TREC

The future of the question answering track at TREC will be guided by both the roadmap document [Burg2000] and the ARDA AQUAINT program[14]. The main changes for TREC 2002 will be the restriction to returning only exact answers as specified by the roadmap, however, due to the poor performance on the context questions this sub-track will be removed and will form part of the first year of the AQUAINT program.

The ultimate goal of the AQUAINT program is not to develop QA systems for only factually based questions whose answers can be found as a single string or within a relatively short window of text (e.g. a 50 or 250 byte window) from a single document[15]. Rather the research intends 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 (basically the top level of application envisaged in the roadmap, see section 2.1 for more details). While some systems exist which offer some results in these areas, they are limited and do not meet the US governments broader requirements for question answering. The major areas of research to be funded are:

- Question Understanding and Interpretation (including contextual interpretation, query expansion, query taxonomy),
- Determining the Answer (including information retrieval and extraction from multiple media/languages and data types, interpretation, synthesis, resolving conflicting information, justification),
- Formulating and Presenting the Answer (including summarization, synthesis, generation), AND/OR
- Cross-Cutting/ Enabling/Enhancing Technologies that directly and materially support the above goals (including but not limited to advanced reasoning, sharable knowledge sources, content representation, interactive QA, role of context in QA, role of knowledge in QA, and language processing and natural language processing research required to support advances in QA.)

The current suggestion is that participants in the AQUAINT program will attempt harder tasks, such as context questions, 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 for all the TREC participants to attempt.

3.5 Other Techniques Relevant to Question Answering

There are many different techniques that could be brought to bear on the problem of question answering. This section aims to outline a few of the most promising of these techniques.

3.5.1 Inference Rules

One of the major problems in question answering is the potential for mismatch between the expressions used in the question and the expressions used in the text. This is the problem which Lin and Pantel address in their paper, Discovery of Inference Rules for Question Answering [Lin2001]. The paper presents an unsupervised

[14] The AQUAINT program can be found at http://www.ic-arda/InfoExploit/aquaint/index.html.
[15] This paragraph is paraphrased from the AQUAINT description that can be found at http://www.ic-arda.org/InfoExplot/aquaint/index.html

algorithm for discovering these inference rules from text. The algorithm is based on Harris' Distributional Hypothesis, which states that words that occur in the same context tend to be similar. Instead of this they use an altered version of the hypothesis, which works with paths in the dependency trees of a parsed corpus, i.e. if two paths tend to link the same set of words they hypothesize that the meanings of the paths are similar.

This method allows phrases that are not the same, but which a read could infer as having a similar meaning, to also be extracted from the text, for example both of the following rules are extracted even though the second one is an inference and the two phrases do not have the same meaning:

> *"X writes Y"* implies *"X is the author of Y"*
> *"X caused Y"* can be used to infer that *"Y is blamed on X"*

They also showed (by experimentation) that people find constructing these inferences manually more difficult then they expected. For example most people would have constructed

> *"X is the author of Y"*

from the text

> *"X wrote Y"*

whereas they would be unlikely to construct

> *"X's Y factory" from "X manufactures Y"*

from the query

> *"What does Peugot manufacture?"*

which can be answered by the following text:

> *"Chrétien visited Peugot's newly renovated car factory in the afternoon".*

The ability of the system to automatically discover these inferences from text allows them to be easily applied to the problem of question answering, especially potential mismatches between the question and answer bearing texts, without vast amounts of time being required to hand-code similar inference rules. Methods like the one described should soon start to appear in numerous different types of NLP systems including those dealing with question answering. This should result in an improvement in the performance of those NLP systems.

4 Progress to Date

The work I have undertaken this year has used, as its basis, the University of Sheffield's TREC system; known as QA-LaSIE, which was entered into TREC 8 and TREC 9 (see [Hump1999] and [Scott2000] respectively for full details of the system).

4.1 Overview of the TREC 9 System

Before discussing how the original system has been updated in preparation for TREC 2002 we will first discuss the layout and workings of the original system, the key features of which are shown in Figure 4-1.

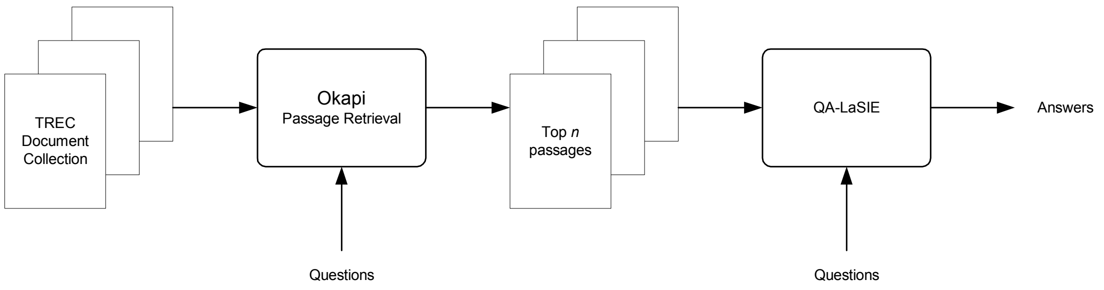


Figure 4-1: Original system setup.

Firstly the TREC document collection is indexed using the Okapi information retrieval system (this is done once only in advance of any questions). This index is then used to return the top *n* passages relevant to the question, the query to Okapi simply being all the question words. The top *n* passages are then submitted along with the question to QA-LaSIE, which should produce one or more answers.

The main work of question answering takes place within QA-LaSIE, which is shown in more detail in Figure 4-2.

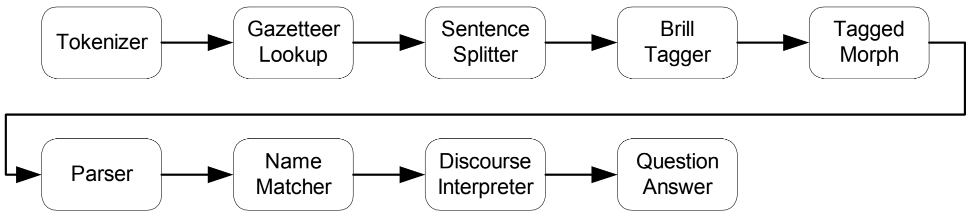


Figure 4-2: Original QA-LaSIE system modules.

The following is a brief description of each of the modules within QA-LaSIE:

- *Tokenizer* – Identifies token boundaries (as byte offsets into the text) and text section boundaries (text header, text body and any sections to be excluded from processing).
- *Gazetteer Lookup* – Identifies single and multi-word matches against multiple domain specific full name (locations, organisations, etc.) and keyword (company designators, person first names, etc.) lists, and tags matching phrases with appropriate name categories.
- *Sentence Splitter* – Identifies sentence boundaries in the text body.
- *Brill Tagger* – Assigns one of the 48 Penn TreeBank part-of-speech tags to each token in the text (see [Brill1992]).
- *Tagged Morph* – Simple morphological analysis to identify the root form and inflectional suffix for tokens that have been tagged as noun or verb.

- *Parser* – Performs two-pass bottom-up chart parsing, pass one with a special named entity grammar, and pass two with a general phrasal grammar. A 'best parse' is then selected, which may be only a partial parse, and a predicate-argument representation, or quasi-logical form (QLF), of each sentence is constructed compositionally.
- *Name Matcher* – Matches variants of named entities across the text.
- *Discourse Interpreter* – Adds the QLF representation to a semantic net, which encodes the system's world and domain knowledge as a hierarchy of concepts. Additional information inferred from the input is also added to the model, and coreference resolution is attempted between instances mentioned in the text, producing an updated discourse model. A representation of the question is then matched against the model, using the coreference mechanism.
- *Question Answer* – Selects the required answer text using the resolved question representation in the discourse model.

Most of this system is unaltered from the LaSIE system entered into the 1998 Message Understanding Conference (MUC-7) which is detailed in [Hump1998]. The changes that were made to create QA-LaSIE were within the Parser and Discourse Interpreter modules and the addition of the Question Answer module.

4.1.1 Question Parsing

Questions were one of the sentence constructions which were not handled by the original LaSIE parser, so extra grammar rules were developed to cover the example questions that were available. The syntactic rules have a semantic component that is used to build up a QLF representation of the question in a similar manner to the rest of the grammar. One major difference between LaSIE and QA-LaSIE is the introduction of a special semantic predicate, `qvar` (question variable), which is used to indicate the *entity* requested by the question. For example, the question *"Who wrote Hamlet?"* produces the following QLF representation:

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

In this representation each entity in the question gives rise to a unique identifier of the form `en`. The use of the word *Who* in the question causes the addition of `person(e1)`, as who suggests the answer to the question will be a person. Also the `qvar` is set to `e1` showing that the question is seeking a person (because `person` and `qvar` share the same entity). The relational predicates `lsubj` (logical subject) and `lobj` (logical object) link any verb arguments founding the text with the verb in the correct relationship.

The QLF representation of the question is stored for use in subsequent processing against the candidate answer texts. When the QLF is stored the entity identifiers are replaced by question entity identifiers of the form `qn`, i.e. `e1` becomes `q1`, `e2` becomes `q2` etc) to facilitate later processing.

4.1.2 Resolution of Question and Candidate Answer Texts

After a candidate answer text has been parsed to produce a QLF representation of each sentence, the QLFs are passed to the discourse interpreter. This behaves exactly the same as in the LaSIE system apart from the addition of a final processing stage.

The discourse interpreter has (by this stage) produced a semantic net of all the entities and relationships present in the multiple QLFs for the document. The net has then had a coreference algorithm applied to it to replace multiple instances of the same entity with a single unified instance (i.e. if `e3` and `e7` refer to the same entity within the text then they will all be replaced by `e3`).

Given this discourse model of a text the QLF of the question is added as the first sentence (`sentence0`) to the model and coreference is then carried out between question entities (`qn`) and entities within the text (`en`).

In the version of QA-LaSIE which was entered into TREC 8 this was the primary QA mechanism: if the `qvar` was resolved with an entity in the text then this entity became the answer; if not, then no answer was proposed. This approach had several major drawbacks. First, it permitted only one answer per question, whereas the QA track allowed five answers and secondly it was very fragile, as coreference was difficult to establish.

Given these weaknesses, the system entered into TREC 9 followed a significantly different approach. Instead of attempting to directly corefer the `qvar` with an entity in the text, entities in the text are scored in a way which attempted to value their likelihood as answers. The best scores were then used to select a single answer to return for each sentence.

1. Each sentence is given a Constraint Score, *C*, equal to 1 point for each Question Constraint that is a member of the sentence, where a question constraint is an entity in the question. This has the effect that sentences which contained entities detected as coreferring with entitles in the question will be rewarded.
2. Within each sentence every remaining entity (`eY`) is tested against the question variable (`qvar`) for:
   a) Semantic Similarity, *S*: the reciprocal of the length of the path between `qvar` and `eY` in the semantic lattice (ontology). For instance if the `qvar` is of type `person` then an entity which also has the type `person` will receive a score of 1.
   b) Property Similarity, *P*: this is between 0 and 1 and is a measure of how many properties the two instances share in common and how similar the properties are (i.e. for a question such as *"Name a green fruit"* where a property of fruit is the colour green it does not make sense to allow the text *"a sweet fruit such as the bright red strawberries"* to be used as an answer because here strawberry has the colour property red).
   c) Object Relation, *O*: 0.25 if `eY` is related to a constraint within the sentence by apposition, a qualifying relationship, or with the prepositions *of* or *in*.

d) Event Relation, *E*: 0.5 if there is an event entity in the QLF of the question which is related to the `qvar` by a `lsubj` or `lobj` relation and is not the `be` event (i.e. derived from a copula construction) and the entity being scored stands in the same relation (`lobj` or `lsubj`) to an event entity of the same type as `qvar` does.

These four values are then added together and then divided by 2.8 to give `eY` a score, which is then added to the sentence Constraint Score, *C*, then divided by the number of question constraints, *Q*, plus one. This can all be viewed as producing the following equation:

$$\text{Score for eY} = \frac{\left(\frac{(S+O+P+E)}{2.8}+C\right)}{1+Q}$$

Equation 4-1: Equation for scoring an answer instance.

4.1.3 Answer Output

The Question Answering module, simply collects the answers from the discourse interpreter (which outputs one per sentence in the document) and ranks them according to their score. Once all the documents for a question have been processed the module simply outputs answers based on the five best scoring entities. The only processing carried out in this module is concerned with selecting either 50 or 250 bytes of text containing the five best answers, no other processing takes place.

4.2 TREC 2001

Although the University of Sheffield's NLP group did not participate in TREC 2001, they stayed involved in the ongoing discussions. As a way of introducing myself to the QA community I volunteered to help produce regular expression patterns which match the accepted answers to the questions used in TREC 2001. These patterns are used to enable researchers to quickly and easily see how their systems perform against the questions, without having to manually judge each answer (which is how the official results for TREC are obtained)[16]. For example question 975 used in the TREC 2001 evaluation was *"When was the first liver transplant?"*. The judgement files published by NIST show all the answers given by all the systems, but without the run tag so there is no way of identifying the system which returned a specific answer. The judgement file is useful, however, to groups who wish to gather questions and answers for a machine learning approach to question answering. A section of the judgement file for question 975 is shown below.

```
975 AP880511-0060 1 in 1967
975 AP880527-0033 1 first liver transplant in 1967 .
975 AP880527-0033 -1 split -
975 AP880527-0033 1 world 's first liver transplant in 1967
975 AP880529-0051 -1 1985
975 AP890104-0148 -1 e left the hospital _Dec. 16_.
975 AP890104-0148 -1 November
975 AP890104-0148 -1 November 1985
975 AP891018-0099 1 1963
975 AP891018-0099 1 [Date:891018] 1963 and pioneered; 1980s
975 AP900118-0221 1 , in 1963, as well as the 31 other liver
```

[16] The patterns along with other QA related files can be found at http://trec.nist.gov/data/qa.html.

The first column is the question number followed by the document identifier. The third column is a 1 if the answer is correct and -1 if it is not. As you can clearly see from this sample all those answers which were judged to be correct contain either the year 1963 or 1967, whereas none of the answers judged to be incorrect contain either of these years. So the following regular expression can be built to match against all the correct answers:

```
196[37]
```

Research groups can then use these patterns to check if an answer they return to the question contains the correct answer or not, without having to manually inspect each single answer.

Since TREC 2001 we have presented the 500 questions from the TREC 2001 competition to our original TREC 9 system. Using the answer patterns mentioned above our 50-byte answer system scored an MRR of 0.169 and failed to find a correct answer to 360 of the 500 questions. Our exact answer system scored an MRR of 0.133 and failed to answer 396 of the questions. If we had taken part these runs would have been ranked approximately 43rd and 51st, respectively, out of the 66 runs that were submitted to the track with the worst system scoring an MRR of only 0.003.

4.3 Changes and Updates to the QA System

Part of the work I have undertaken this year has been to migrate the QA system from the original GATE (General Architecture for Text Engineering) framework, [Gaiz1996], in which it ran during TREC 8 and TREC 9, to the new GATE 2 framework (see [Cunn2002a] and [Cunn2002b]). This has meant a large amount of re-writing of the wrappers for the components used, while leaving the components themselves largely unchanged, i.e. the bottom-up chart parser and the discourse interpreter are virtually identical to those used in TREC 9. This has allowed me to familiarise myself with the system ready for the more detailed work to follow. The new layout within GATE 2 of the question answering system can be seen in Figure 4-3, note that those modules marked by an asterisk are available as part of GATE 2 and are used unmodified in the question answering system.

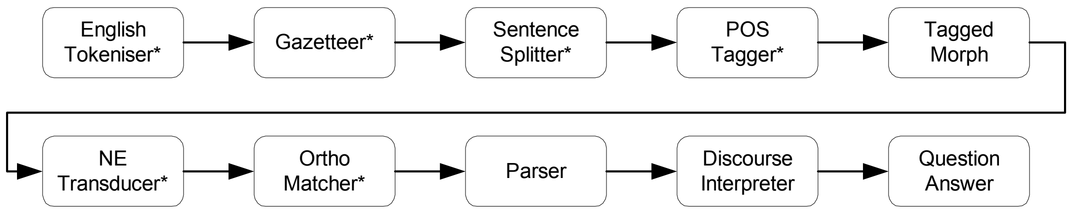


Figure 4-3: The layout of the QA system in GATE 2.

The main changes to the layout are the use of a different POS tagger (Heptag, see [Hepple2000], instead of Brill's tagger) and the fact that two modules, NE Transducer and OrthoMatcher, have replaced the Name Matcher component. These changes, however, make no significant difference to the way in which the question answering system works.

The only changes to the bottom-up chart parser are the addition of a few extra grammar rules aimed at increasing the performance of parsing specific types of question. The following sections outline the changes made to the other modules in preparation for TREC 2002.

4.3.1 Processing Answer Instances

The discourse interpreter has seen some alterations to make answering scoring more appropriate. Processing, of a document, is the same up until the point at which the possible answers are identified and then scored. The previous system simply returned the highest scoring answer for each sentence within a document, i.e. the total number of possible answers was equal to the number of sentences within the document. This has a number of problems associated with it:

1. Often more than one answer within a sentence is awarded the same score. If this happens for answers with the highest score then the system can only return one of the possible answers, and therefore always returns the one that appeared last in the sentence (i.e. the one it processed last).
2. Often the answer module (which is completely independent of the discourse interpreter) will throw away an answer as it is deemed unlikely to be correct. If this happens then a sentence will not have an answer associated with it (due to point 1), and on occasions this might result in a document having no answers associated with it.

These two issues are both addressed by simply returning all possible answers and their associated scores from the discourse interpreter and allowing the answer module to decide which to make use of and which to discard.

4.3.2 Property Similarity

When it comes to scoring answers alterations have also been made to the previous system. One of the components of the score for an answer is property similarity, which was mentioned in section 4.1.2. In the TREC 8 system all common properties between `qvar` and `eY` had to match for `eY` to be considered a possible answer. This, however, was deemed to be too hard a constraint and was relaxed, for TREC 9, to the number of common matching properties of `qvar` and `eY`, divided by the total number of properties of `qvar` and `eY`. From analysis of the system this appeared to still be too strict a requirement and hence property similarity has been removed altogether.

4.3.3 Semantic Similarity

The major addition to the scoring algorithm is concerned with the semantic similarity between `qvar` and `eY`. In the TREC 9 system the semantic similarity is defined as the reciprocal of the distance between `qvar` and `eY` in the systems ontology (the path was not even constrained to the shortest possible path, just the first to be found). The problem with this approach is that the ontology is extremely small, and so often `qX`, `eY` or both are not in the ontology and hence get a semantic similarity score of zero, even if they are clearly linked in some way (i.e. neither *house* nor *abode* are in the ontology although these two words are clearly related). The solution to this has been to implement a two-stage process. Firstly the original algorithm (with the added constraint of always returning the shortest path) is used and only if no path is found is the new method used. This allows us to specify specific relationships in our ontology if we deem them necessary.

The new method takes WordNet (see [Mill1995]) and assumes that it is an ontology. The semantic similarity of two entities, `qvar` and `eY`, is then computed using a variation of the Leacock and Chodorow method (LCH) presented in [Lea1998]. In their original method semantic similarity is defined in Equation 4-2.

$$\text{Semantic Similarity} = -\ln\left(\frac{d}{32}\right)$$

Equation 4-2: The Leacock and Chodorow Semantic Similarity equation.

Where d is the distance between the two words in question[17]. The distance between two words is calculated by building hypernym (… is a kind of …) trees, one for each of the noun senses of both words. These trees are then overlapped and the distance between two words is the number of hypernym (or hyponym if going down a tree) relationships required to go between them, plus one. As an example, assume we want to know how semantically similar the words *fish* and *food* are to each other. Firstly we build the hypernym trees for the all the noun senses of both words, these can be seen in Figure 4-4.

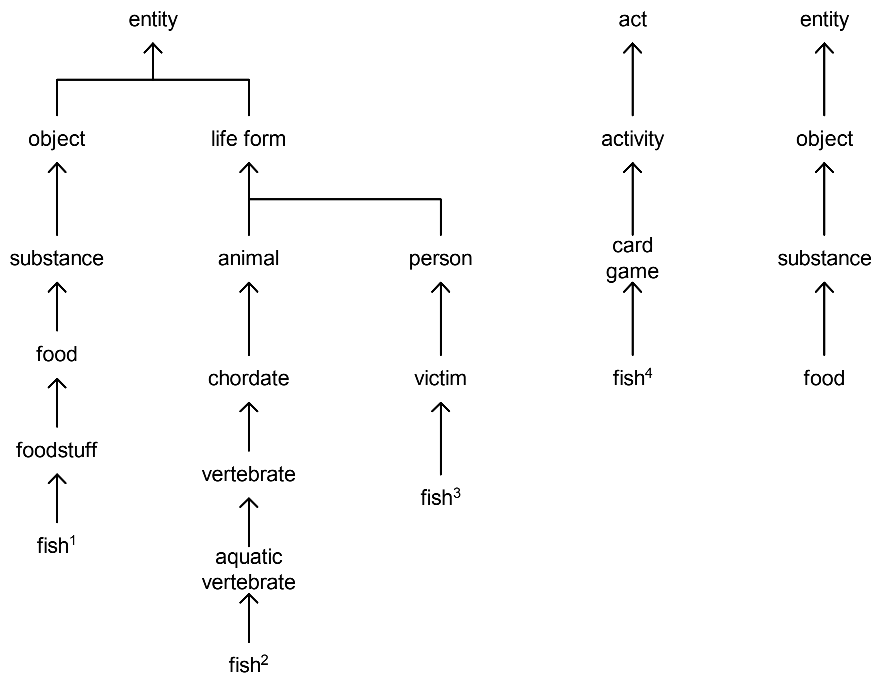


Figure 4-4: The hypernym trees for *fish* and *food*.

We then work out all the paths between fish and food using the generated hypernym trees. It turns out there are three distinct paths (in which ↗ signifies a hypernym relation, ↘ signifies a hyponym relation and = signifies that that two things are identical, i.e. the join between two hypernym trees):

1. fish^1 ↗ foodstuff ↗ food = food
2. fish^2 ↗ aquatic vertebrate ↗ vertebrate ↗ chordate ↗ animal ↗ life form ↗ entity = entity ↘ object ↘ substance ↘ food
3. fish^3 ↗ victim ↗ person ↗ life form ↗ entity = entity ↘ object ↘ substance ↘ food

[17] Note that the normalising factor of 32 is not arbitrary but is double the maximum depth of the WordNet hierarchy.

The shortest of these three paths is shown more clearly in Figure 4-5.

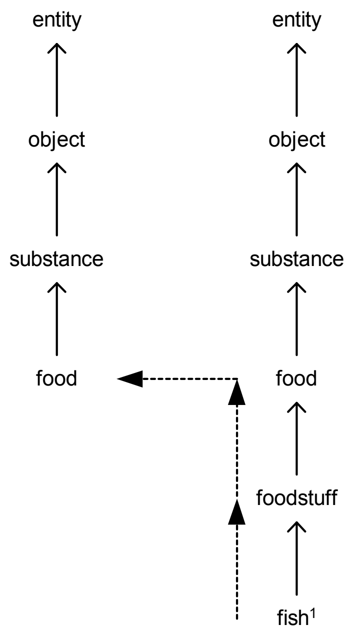


Figure 4-5: The shortest path from *fish* to *food*.

Equation 4-2 for calculating the semantic similarity does not produce a score, which is in anyway comparable to the semantic similarity already calculated by our system using the built-in ontology. To correct this problem we use the equation given below to calculate semantic similarity.

$$\text{Semantic Similarity} = \frac{1}{d}$$

Equation 4-3: Equation used to calculate semantic similarity in our system.

Table 4-1 presents the distances, LCH measure and our measure for each of the paths.

Path Number	*Distance*	*LCH Measure*	*Our Measure*
1	3	2.37	$\frac{1}{3}$
2	10	1.16	$\frac{1}{10}$
3	8	1.39	$\frac{1}{8}$

Table 4-1: Table showing distances and scores for the three paths.

Other possible methods of computing the semantic distances between words, using WordNet, could have been used and these include [Jiang1997], [Resnik1995], [Lin1998] and [Hirst1998].

4.3.4 Answer Scoring Algorithm

The modified answer scoring algorithm is identical to that detailed in section 4.1.2, apart from the complete removal of the property similarity component, which gives the following equation:

$$\text{Score for eY} = \frac{\left(\frac{(S+O+E)}{2.8}+C\right)}{1+Q}$$

Equation 4-4: Equation, used in TREC 2002, for scoring an answer instance.

4.3.5 Question - Answer Overlap

The one component of the system that is radically different is the final answer module. This takes the possible answers as identified by the discourse interpreter and ranks them according to their score and other attributes. Before the algorithm for ranking the answers can be explained a few minor algorithms and ideas, which it uses, have to be covered.

As has already been discussed, in section 3.4, window-based methods for pinpointing answers are severely limited and are unlikely to be involved in the future of QA research. Our system goes one step further than this, however, and assumes that overlap between the question and a candidate answer is inherently bad. Clearly for a question such as *"Where is Perth?"* an answer of *"Perth is in"* is not correct and can be eliminated using the following method.

In most cases it is unlikely that a correct answer to a question will contain many, if any, of the non-stopwords in the question. We can use this assumption to throw away some of the possible answer strings before we even look at the score assigned to them. Word overlap between a question and candidate answer is best viewed as a percentage. At 0% there is no overlap between the question and candidate answer and so the string may be a correct answer to the question and therefore requires further processing. At 100% overlap all the non-stopwords in the candidate answer appear in the question, at which point it is highly unlikely that this string will be a correct answer to the question and can then be disregarded (an exception is TREC 2001 Q1026 *"What does target heart rate mean?"* which has as one of its possible answers *"target heart rate"*, although the more important question here is whether *"target heart rate"* is in fact a valid answer to the question). At points in between it is unclear whether the candidate answer may be correct, or not, based only on the percentage overlap.

The initial attempt at including overlap in the system worked simply by assuming that if there were any overlap at all then the candidate answer would be discarded. As a naïve approach this was actually quite successful, however, the current system simply discards any answers where the overlap with the question is 100%.

4.3.6 Combining Semantically Similar Answers

Having carried out some limited analysis of the performance of our system over the TREC 2001 questions, one thing was clear; we would often return two or more semantically equivalent answers. Clearly if the answer is correct then this is alright, but if these answers are wrong then this may well prevent correct answers from appearing in the top *n* documents which we are allowed to return. On some occasions we were actually returning identical answers (i.e. for Q1000 we returned five answers all of which were *"the sun"*), these are easy to remove by simply keeping only the highest scoring of two identical answers.

Unfortunately, equivalent answers are not always identical strings; as is the case for the question *"Where is Perth?"* to which our system returned within the top five answers: *Australia* and *Western Australia*. Clearly *Australia* and *Western Australia* are semantically equivalent answers to the question, so only one of them need be returned.

The approach taken to deal with these answer strings, which is similar to that used in [Brill2001], is to test if two answers A and B are the same by checking that the stem of every non-stopword in A matches a stem of a non-stopword in B, or vice versa. Using this test, if two answers match, then both are removed and a new answer is created from the highest of the two scores and the longest answer string. The effect of this method on our example question was that now only *Western Australia* is listed as one of the top five possible answers.

Clearly the same approach to the question *"In which country is Perth?"* would not be as effective as *Western Australia* is not an exact country name, this method is still better than simple string matching approaches although it still needs some improvement.

Using this approach improved the system performance slightly. More importantly was the unexpected side effect which caused the system to clarify some answer strings, with the most obvious being people's names: '*Armstrong*' becoming '*Neil A. Armstrong*' and '*Davis*' becoming '*Eric Davis*'.

4.3.7 Strategy for Ranking Answers

Using the ideas outlined above for refining the list of possible answers we can now detail the algorithm used to rank the answers for a specific question.

1. Firstly a NIL answer is added to the list of possible answers, *LA*. Currently this answer is given a score of zero, although through further work it may be possible to give this answer a score - hence introducing a cut-off point below which a score is so small that the chances of the answer being correct are minimal.
2. We then calculate the percentage word overlap between the current document and the question (ignoring stopwords). From experience we know that the performance of our IR step is not as good as we would like, so this calculation has the effect of re-ranking the documents returned by the IR system. Hopefully when more work has been carried out on the IR system we will be able to remove this step altogether.
3. For each candidate answer:
    - We get the score, the answer text, the rank of the document it came from and if it is an exact answer or not (exact answers come from `name(eX,Y)` in the semantics, other answers are snippets of text taken from the documents).
    - If the overlap between the answer and the question is 100% then this answer is discarded.

- If the score for this answer is greater than any answer we have seen thus far, within this document, then we store all the other answers to date, from this document, in a list, *LO*, and create a new answer list, *LP*, with just this answer. If it is the same as the currently highest scored answer then we put them both in the list *LP*. If it is lower than the currently highest scored answer we put it straight in the list, *LO*.

4. We now take the list of highest ranked answers, *LP*, and add these to the set of answers taken from previous documents, *LA*. This process of adding them in makes sure semantically similar answers are merged rather than simply added.
5. We then take the list of all the answers from all the documents processed so far for this question, *LO*, which were not the highest ranked answers and merge these answers with the list we updated in step 4, *LA*. The difference here is the answers are not added but only merged, i.e. scores are only updated if an answer in the list *LO* is semantically the same as one in the highest ranked list, *LA*, but has a higher score.
6. Once all of the documents for a question have been processed in this way then the list of answers, *LA*, is sorted on the following attributes (i.e. if the value of the first attribute is the same for two answers then we sort on the second attribute, two answers are equally ranked if all of their attribute values are identical):
   - The score (the higher the better)
   - The question-document overlap (the higher the better)
   - The number of other answers which were semantically the same as this one (the higher the better)
   - The rank of the document from which the answer originates (the lower the better)
   - If the answer is exact or not (exact is better).
7. The number of answers requested by the user is then taken from the top of the *LA* list and returned.

4.4 Boosting System Performance Using Answer Redundancy

As has been reported by numerous research groups, including [Light2001], the number of answer instances (within a single document or multiple documents each containing the answer once) directly impacts the end-to-end performance of a QA system. This is partly due to the fact that the IR engine is more likely to find a relevant document, and also because there may be multiple different wordings of an answer within the text; giving the parser's grammars a better chance of getting at least one of them to parse in a way that is beneficial to the rest of the system.

To this end it was decided to attempt to boost the knowledge available to our system, not as may be expected, by returning more documents at the initial IR step, but by using two different text collections. The second text collection that was chosen was the World Wide Web. A document collection for a single question is made up of the snippets displayed on the Google results page for the top ten documents returned by Google. These are certainly not full documents, and are rarely full sentences but this is not a problem as the bottom-up chart parser we employ simply returns the best parse, it is not constrained to only returning a full sentence or a complex phrase. This method of using just the snippets has been shown to be successful in [Buch2001], although they used the snippets from the first one thousand documents rather than the first ten.

The QA system is run against both text collections and then the results are merged together. The end result must be an answer which references a document in the TREC collection so the process of merging is as follows: for each answer returned from the Google corpus (both the list of high scoring answers and the list of rejected answers), if an answer exists from a document in the TREC corpus which is semantically equivalent, then merge by keeping the highest score etc., but the reference to the TREC document.

Over a sample of one hundred questions (TREC questions 1000 to 1099) the results of combining the collections in this way were (based on returning the top five answers for each question):

Collection	*MRR*	*Not Found (%)*
TREC	0.256	68 (68%)
Google	0.227	68 (68%)
Combined	0.285	65 (65%)

Table 4-2: Results of using Google to boast system score.

Using multiple document collections, especially the WWW, has been previously reported in Data-Intensive Question Answering by Brill et al [Brill2001]. Their reasoning behind using the web is the same as ours; to increase the number of answer instances within the text available to process. Their system differs vastly from ours, however, in the way in which they make use of this extra collection. Their system finds the top five answers from the WWW using Google as the information retrieval system (the method of finding answers is immaterial for this comparison and is therefore not detailed here). At this point for each of the top five answers they submit a query to the TREC collection (indexed using Okapi) consisting of the question words and the candidate answer. The top ranked document is then returned as the supporting document for the answer. The difference between this system and ours should be clear; Brill et al make no attempt to process the documents in the TREC collection and rely solely on the answers obtained from the web. This seems inherently dangerous as there is no way of knowing if the top document returned from the TREC collection for a query and candidate answer, relates in any way to the question being asked. Fortunately, it appears that their system for finding answers from the web is highly reliable and hence the chance of one of the answers having a supporting document assigned to it is quite high. At TREC 2001 the system scored an MRR of 0.347 (in two separate runs), which ranked them 12th and 13th out of the 66 runs submitted to the main task.

4.5 List Questions

List questions are inherently harder to answer than standard, single answer questions, mainly because the systems have to combine information from multiple sources to locate the required number of answers. Also the system has to be able to extract from the question the number of different answers required.

Our simple solution to these problems is as follows. The system processes the question in the usual way producing a long list of answers (sorted in the same way as before). The question is the scanned, token by token, until the first token whose part of speech (POS) signifies that it is a number, we then assume that this is the number of answers sought, which are simply returned from the top of the answer list already produced.

Clearly this suffers from the obvious problem that some questions may contain more than one number, i.e. *"The United States has 2 main political parties name 1 of them."*

Our system appears to have another flaw, in that the grammar used by the chart parser was not designed to handle questions such as *"Name 20 countries that produce coffee"* and so often fails to instantiate a `qvar` which means it is very unlikely the system will be able to generate a correct answer (as by default the `qvar` is associated with the first element in the semantics, which is usually `question`, due to the **question** which is pre-pended to every question). To alleviate this problem, if it is known that the question being processed is a list question and a `qvar` has not been instantiated then the semantics are searched for the first instance of `count(eY, Z)` and then the `qvar` is initialised to `eY` (note that `Z` should be the number of answers we are looking for but this is not currently checked). So for the example question the semantics would contain "`countries(e2), count(e2,20)`" hence "`qvar(e2)`" will be added to the semantics, successfully initialising the `qvar` to the correct type. In a real system this would not be possible as there would be know way of knowing in advance that we were processing a list question, however in TREC the list track is completely separate to the main track so we know before the run starts that we are dealing exclusively with list questions.

4.6 A Framework in which to Develop Grammars

One of the main areas of the system that still requires some work is the grammar used in the bottom-up chart parser. Many books try to express the English language in terms of grammar rules (see [Gee1983], [Jarv1993] and specifically [Burt1997]), it is however, very difficult to develop grammars, even from these guides, as a small change in one rule, to fix a phrase which has not parsed correctly, will often have many little knock-on effects in other phrases.

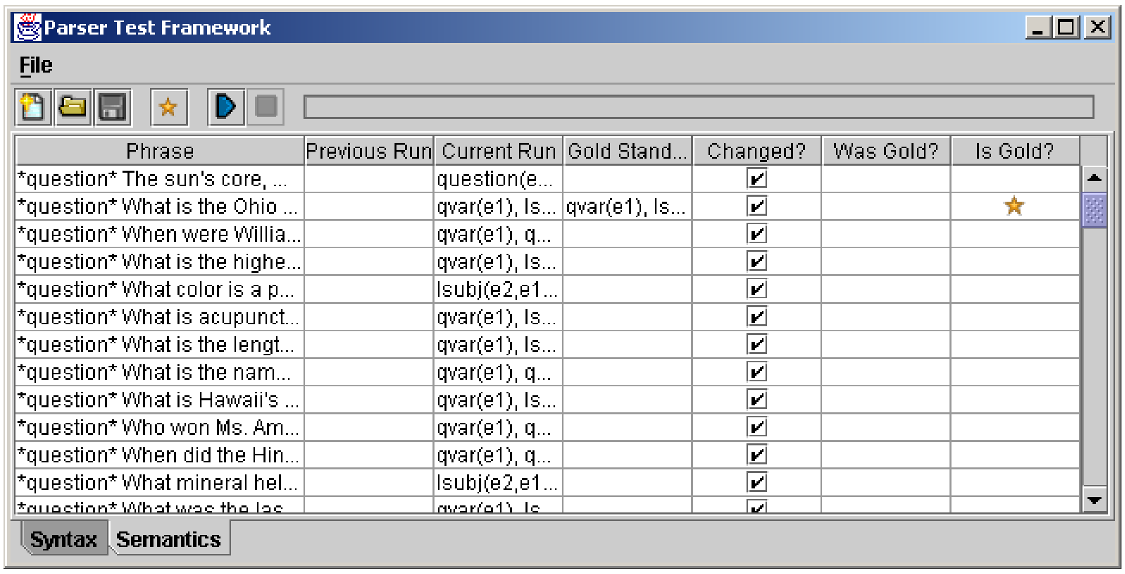
Therefore, the best way of developing grammars is to have a test framework in which one can quickly see all the knock-on effects of a change over numerous different phrases.

To this end I have developed an application, using GATE 2 as the processing engine in the same way as the QA system, which will allow grammars to be developed and tested in an easy way[18]. The interface to the application can be seen in Figure 4-6.

This framework is designed to allow the user to develop a grammar and quickly see the changes a new rule makes to both the generated syntax and semantics for a number of phrases. To this end the syntax and semantics are presented in two different tables (accessed through the tabs at the bottom of the interface). Both tables are identical in contents and allow the user to see; the phrase being parsed, the result of the previous parse, the result of the current parse, the gold standard parse (set by the user) whether or not the current parse is different to the previous and whether or not the previous or

[18] The framework application can be used by members of the department by executing the script `/share/nlp/projects/trec11/parser_test/framework/run.sh` and the grammar in use can be found in …`/parser_test/buchart/grammar`.

current parses were identical to the specified gold standard parse. This is a lot of information but it is presented to the user in a way which any changes are quickly visible.



Phrase	Previous Run	Current Run	Gold Stand...	Changed?	Was Gold?	Is Gold?
question The sun's core, ...		question(e...		✓		
question What is the Ohio ...		qvar(e1), Is...	qvar(e1), Is...	✓		★
question When were Willia...		qvar(e1), q...		✓		
question What is the highe...		qvar(e1), Is...		✓		
question What color is a p...		lsubj(e2,e1...		✓		
question What is acupunct...		qvar(e1), Is...		✓		
question What is the lengt...		qvar(e1), Is...		✓		
question What is the nam...		qvar(e1), q...		✓		
question What is Hawaii's ...		qvar(e1), Is...		✓		
question Who won Ms. Am...		qvar(e1), q...		✓		
question When did the Hin...		qvar(e1), q...		✓		
question What mineral hel...		lsubj(e2,e1...		✓		
question What was the las		qvar(e1), Is		✓		

Figure 4-6: An application for assisting the development of grammars.

4.7 TREC 2002

Having made the changes outline above we took part in TREC 2002 submitting a total of five runs, three in the main track and two in the list sub-track. The different runs were as follows:

- `sheft11mo3` – This main run used only passages of up to three paragraphs retrieved from the AQUAINT collection using Okapi.
- `sheft11mog3` – This main run used passages of up to three paragraphs retrieved from the AQUAINT collection using Okapi, and also the top ten snippets returned by Google for the question.
- `sheft11mog1` – This main run was identical to `sheft11mog3` apart from the maximum passage size was limited to only one paragraph.
- `sheft11lo` – This list run used the same settings as `sheft11mo3`.
- `sheft11log` – This list run used the same settings as `sheft11mog3`.

Once the runs had been submitted to TREC for evaluation I started to carry out some simply evaluation and failure analysis to see roughly how we had fared over the 500 questions.

4.7.1 Broken Questions

A *Broken Questions* is one that causes the system to fail for some reason. There were a number of different reasons that caused questions to be classified as broken during TREC 2002:

1. *Chart Parser Problems* – This problem occurred only if a document was very long or if it had a very odd structure, the problem being that the question never completed processing (one of the questions was stopped after it had been processing the same document for 68 hours). The majority of documents that caused this problem were lists of football matches (i.e. each line was ‘`CLUB vs`

CLUB') and no sentence breaks were found in the document. This meant the entire document, with its odd structure, was treated as a single sentence. As yet no attempt has been made to fix this issue, although it is likely that the easiest option would be to modify the sentence splitter to introduce more sentence breaks in this type of document.

2. *WordNet Access Problems* – The code used to access WordNet, works by carrying out numerous binary searches over the index and data files to locate the required information. This is much faster than having to compile and load the entire WordNet database into Prolog for each document. When accessing the database, however, the index and data files were being repeatedly opened and closed; unfortunately the closing of the files does not appear to release the operating system file handles. This meant that for large documents, which necessitated a vast number of WordNet look-ups, the operating system would refuse to allow one of the files to be opened again which meant that the discourse interpreter would fail, losing all possible answers contained within the document. This problem has been fixed by simply opening each file once (the first time it is accessed) and then seeking to the beginning of the file on future occasions. Unfortunately this does have the unfortunate side effect of slightly slowing down the processing of each document.

4.7.2 Other Bugs in the System

So far a few small issues have been identified with the system, which may have affected the performance of the system over the TREC 2002 questions. Some of these were covered in the previous section but those that did not cause the system to fail are documented here.

1. *Case Sensitivity* – The grammar rules are case sensitive but this is usually not an issue as the attribute of a word, which is usually used, is referred to as the `m_root`, which stores the morphological root of the word as calculated by the Tagged Morph module. This attribute should always be stored in the lower case form and this usually is what happens. Unfortunately a bug in the new Tagged Morph wrapper meant that words for which a root is not known (usually because the word is already the root) is stored in the same case as it appears in the text. This caused a serious problem with the list questions as most of them start *"Name…"* and *Name* is already the root of the word and so was placed into the chart parser with a capital letter so the question rules referred to *name* and they did not match against the `m_root` of *Name*, hence a lot of the list questions were not correctly interpreted leading to worse results than should have been obtained.

4.7.3 Answer Ranking

A quick look at the submission files for the main task shows one thing quite clearly. Although the system is relatively good at answering questions the algorithm for ranking the answers to multiple questions is most certainly lacking. As was explained in detail in section 4.3.7 the ranking algorithm uses each attribute of an answer in a set sequence in order to rank the answers for a question and in the same way to rank answers for multiple questions. The problem which arises is as follows: a question may be incorrectly answered with an answer which scores very highly for some reason but is only seen once and a another question could be answered correctly by a low scoring answer which we see multiple times. For example the two questions *"What is* *the appropriate gift for a 10th anniversary?"* and *"Who is Tom Cruise married to?"* (1608 and 1395 from the TREC 2002 set) have the following entries in the `sheft11mog1` submission file (the format is explained in section 2.4.1).

```
1608 sheft11mog1 XIE19961211.0078 Zhebung Temple Celebrates 1st
1608 sheft11mog1 0.8616071428571429 100.0 3 1.0 true
…
1395 sheft11mog1 APW19990612.0066 Nicole Kidman
1395 sheft11mog1 0.35714285714285715 100.0 54 1.0 true
```

Clearly the answer to question 1608 is wrong, we do not even have to look at the supporting document to know this, however it is ranked sixth in the submission file because of its very high score (0.861607) even though it is only seen 3 times in all the documents processed. On the other hand the answer to question 1395 is right (or at least it was when the supporting document was written, see section 3.4.2 for a discussion of temporal questions) but it is ranked four hundred and twelfth in the answer file due to its relatively low score (0.357143) even though the answer is seen 54 times. Clearly this answer should be ranked a lot higher in the submission file. What may not be instantly obvious is if this problem occurs when ranking the answers to multiple questions surely it will also occur when ranking the multiple answers to a single question. A brief experiment was carried out over 100 of the TREC 2001 question (Q1000 to Q1099) to see if changing the ranking algorithm to use the score multiplied by the number of occurrences would improve the performance within a single question. The results are shown in Table 4-3.

System	*MRR*	*Not Found (%)*
Original Ranking Algorithm	0.288	61 (61%)
Score * Number of Occurrences	0.343	59 (59%)

Table 4-3: Results of combining score and number of occurrences.

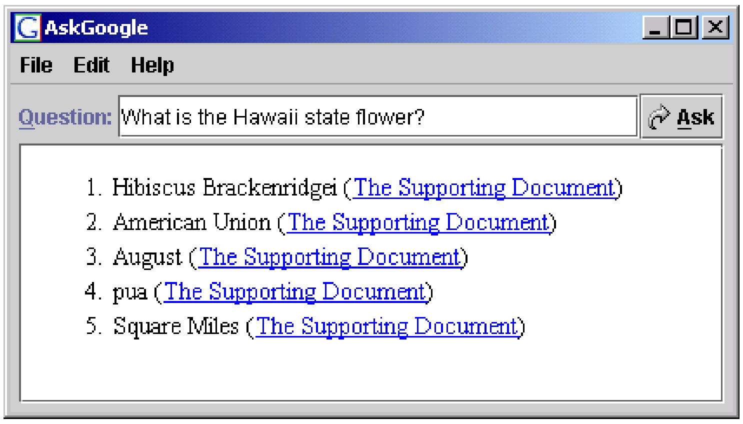
Although the difference is not vast it should be clear that it is still a significant improvement over simply using the score as the main ranking attribute. More work needs to be carried out to see if this naïve approach of simply multiplying the score by the number of occurrences is the best method of ranking answers or whether there is a better alternative.

4.8 Question Answering over the World Wide Web

The original idea for adapting our TREC 9 QA system to use the WWW as its document collection was outlined in [Bamf2001]. Unfortunately, this project was never successfully completed - the web front end and the QA system were never integrated.

More recently, before the QA system was moved to GATE 2, the project was restarted using the original specification. This system worked by getting Google to return the top 100 related sites using the full question text as the search query. The top *n* documents (where *n* is between 1 and 10 and can be specified by the user) which Google lists as being less than 6K in size were downloaded to produce the document collection for the question. The size restriction was an attempt to allow the processing of the question to be completed within a reasonable amount of time. The answers produced were then formatted for display in the user's browser with links to the documents from which they were taken[19]. Unfortunately the system is restricted in numerous ways, especially the restriction to small documents and the instability of the old GATE system. As the GATE 2 framework is more reliable, development of this specific interface has been halted.

An improved WWW QA system using GATE 2 has been developed which takes advantage of the Web APIs, which Google has made available for applications to access the search engine using SOAP[20]. The system uses the top 10 snippets returned by Google, for the question words, to build the document collection over which the question answering system can then run. Although the system only uses the top 10 snippets, which rarely consist of full sentences the accuracy of the system is comparable to the system running on the top 20 relevant documents from the TREC collection. The main reason for limiting the number of snippets to 10 is to allow the system to process a question in a relatively short period of time. If time was not an issue then the system would have been designed to use the top 100 or even top 1000 snippets as other groups have done.



What is the Hawaii state flower?

1. Hibiscus Brackenridgei (The Supporting Document)
2. American Union (The Supporting Document)
3. August (The Supporting Document)
4. pua (The Supporting Document)
5. Square Miles (The Supporting Document)

Figure 4-7: The interface to our new WWW QA system.

At the moment, although the application is functional, there is still work to be done to produce a user-friendly system[21]. It has been suggested that this system may in fact be integrated with another project to allow spoken questions to be answered over the WWW.

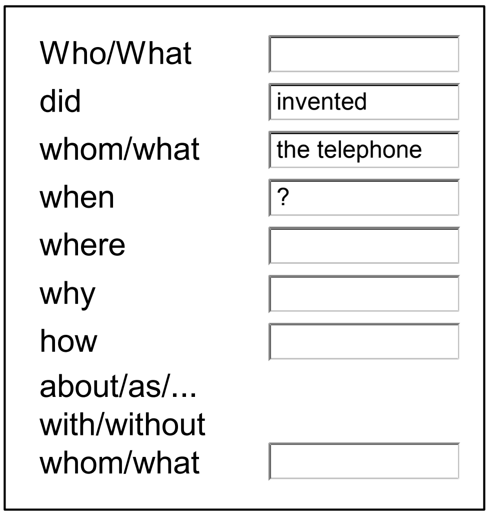
4.8.1 Comparison to Other WWW QA Systems

Probably the best-known question answering system that works over the WWW is Ask Jeeves[22]. Little information is available on exactly how Ask Jeeves works, [Ask2002] contains what little they are willing to admit. The claims are that Ask Jeeves carries out syntactic and semantic processing of the questions which is then used by a template response system to provide a list of more detailed questions. When a user selects one of the more detailed questions a proprietary knowledge base, containing

[19] The online QA system can be found at http://raki.dcs.shef.ac.uk. Access is restricted to machines within the University of Sheffield's Department of Computer Science.
[20] The Google Web APIs are available for download from http://www.google.com/apis/.
[21] The new web QA system can be used by members of the department by executing the script `/share/nlp/projects/trec11/AskGoogle/run.sh` although each user will require a Google Web API licence key which can be freely obtained from http://wwww.google.com/apis/.
[22] Ask Jeeves can be found at http://www.askjeeves.com.

answers to more than seven million questions, is used to provide the answers to the user. Clearly Ask Jeeves differs from our question answering systems in many ways, but basically the fact that we process each question from the same initial state whereas Ask Jeeves uses its knowledge base (which is at least partially hand constructed) to answer questions and updates the knowledge base when asked a question which it has not encountered before.

One WWW question answering system, SHAPAQA[23] presented in [Buch2001], is much more like our system then Ask Jeeves. The main difference between this system and ours is that they have bypassed the problem of having to understand the question. To ask a question using SHAPAQA, a user simply fills in a web form similar to that shown in Figure 4-8, which asks them for certain *phrases*.



Who/What	
did	invented
whom/what	the telephone
when	?
where	
why	
how	
about/as/... with/without whom/what	

Figure 4-8: SHAPAQA User Interface.

This interface works by the user entering the parts of the question they know (the *given phrases*) and placing a question mark against the *phrase* they are looking for. As can be seen from the example given above, which shows how to fill in the form for the question *"When was the telephone invented?"*. The rest of the QA system is then similar to ours with syntactic and semantic processing of the documents taking place in order to attempt to answer the question.

Another online question answering system is MULDER[24] developed by Kwok et al., which they claim to be the first general-purpose, fully-automated question-answering system available on the web (see [Kwok2001]). This system works in a very similar way to ours; the user enters a question in English and then the answers are displayed in the browser with links to the relevant documents. The only real difference is that with each answer the system also gives a measure of how confident it is in the answer, a feature which it would be nice to incorporate in our system. Their major claim (other than being first) is that using just Google requires a lot more user effort to achieve the same level of recall as MULDER.

[23] SHAPAQA can be found http://ilk.kub.nl/~antalb/abvi/week3/shapaqa.html.
[24] MULDER can be found on the web at http://mulder.cx.

5 Issues and Challenges Facing Question Answering

There are many challenges which modern question answering research must address, this section outlines some of the more important issues and asks where we can go in search of solutions.

5.1 How Important is Reliable Information Retrieval?

Most modern QA systems consist of two *almost* separate stages, an initial IR stage and then a further stage (be that semantic processing or pattern matching) that extracts the answers from those documents/passages deemed relevant by the IR engine. This has a serious implication: if the IR stage does not return relevant documents then no amount of processing will result in a correct answer being extracted.

In an experiment using the 432 question from TREC 2001 for which at least one system found a correct (non NIL) answer our IR stage retrieved at least one relevant document for 347 of the questions or in other words we found a relevant document for 80.32% of the questions. On first sight these figures seems quite reassuring, that is until you realise that we failed to find a relevant document for roughly 20% of the questions, i.e. no matter what we do we are limited to a maximum MRR of 0.8. Yes, an MRR of 0.8 is way above the current ability of the system (or most systems come to think about it), but it is still a limiting factor and will clearly prevent any system using this as its basis from ever answering every question presented to it, for which the text collection contains an answer.

An important question involving information retrieval is the level (i.e. the amount of text returned) at which retrieval should take place. The reason for needing IR is that no NLP based system could ever hope to process an entire collection to find the answer to a question, so IR acts as a filter between the document collection and the QA system only allowing documents which are relevant to be processed. The question, however, is how far we can narrow the filter: do we only allow through whole documents or is it better to allow through short paragraphs or even just sentences?

One of the worries of returning short passages or sentences is anaphora resolution. The passage returned by the IR engine as relevant (i.e. contains most of the query terms) may not actually contain the answer, rather a neighbouring passage may refer, through words such as he, she or it, to the query terms. In this scenario although a passage from a relevant document has been returned the answer is not and the chance of this occurring is increased the smaller the returned passage becomes. On the other hand returning whole documents increases the amount of processing the NLP system has to perform and can introduce noise by providing more entities within the text which may obscure the actual answer to the question. As an example, consider the question *"When was Mozart born?"* for which a document containing the following is returned by the IR engine: *"Mozart was a composer of many well known pieces of classical music. He was born in 1756."*. Now it is clear that depending on the query words given to the IR system there is the possibility that the second sentence may not be returned if we limit the system to returning only a sentence, where clearly we will need both sentences to answer the question: the first to know we are dealing with *Mozart* and the second sentence to give us the birth date of a man (and as there is only one man mentioned) who must be *Mozart* (hence the need for the first sentence).

Numerous experiments were carried out on the different sizes of passage used as the document collection for our QA system (these experiments are documented in [Rob2002]). The main results from these experiments are detailed in Table 5-1, in which the best results in each column are underlined for clarity.

Passage Length	*%ABD*	*IR MRR*	*Correct Answers (out of 100)*	*TREC Score*
1 paragraph	67%	*0.3354*	<u>13</u>	0.2068
2 paragraphs	<u>74%</u>	*0.4096*	11	<u>0.2097</u>
3 paragraphs	72%	*0.4025*	7	0.1631
4 paragraphs	72%	*0.3925*	8	0.1559
5 paragraphs	70%	*0.4203*	7	0.1542
6 paragraphs	70%	*0.4249*	7	0.1511
7 paragraphs	71%	*0.4566*	7	0.1588
full documents	72%	*<u>0.4800</u>*	7	0.1683

Table 5-1: Results of IR experiments and their effects on the QA system.

The definition of the data in each column is as follows:

- *%ABD* - the percentage of questions for which at least one relevant answer bearing document was found in the retrieved data.
- *IR MRR* - a mean reciprocal rank measure for the IR performance: the mean of the reciprocals of the rank of the first relevant answer bearing document retrieved by Okapi, or 0 if no such document was retrieved. These two measures are computed using the relevance judgements and Perl answer patterns supplied by NIST.
- The number of correctly answered questions (out of 100), i.e. the number of questions for which the exact answer returned by the system matched one of the Perl patterns supplied for that question.
- The TREC-2002 style score for the run (see section 2.4.1).

From these results it is unclear that there is any significant difference between using documents of one, two or three passages in length, although clearly returning a small number of passages (three or less) produces significantly better end-to-end results (there is a small statistical advantage to using passages of three paragraphs in length).

The IR step of our system only returns the top twenty documents which match the query. This is to allow the system to process the documents in a reasonable amount of time and because supplying the system with large amounts of text cause it to be more unstable (mainly through issues of memory). This does, however, mean that some of the relevant documents are being disregarded before the NLP modules have a chance to work on them. To find out what effect returning only twenty documents had on the %ABD a series of experiments were carried out which are summarised here (detailed results can be found in [Rob2002]).

All 1193 questions from TRECs 9 and 10 were used to calculate the %ABD at numerous different cut-of points (5, 10, 30, 30, …). Of the 1193 questions no answer was found for 93 of them and so the maximum %ABD value that can be obtained is 90.02%. The results of these experiments using documents of 1, 2 and 3 paragraphs in length are shown in Table 5-1, and presented graphically in Figure 5-1.

Doc Length (paragraphs)	*%ABD at Rank*								
	5	*10*	*20*	*30*	*50*	*100*	*200*	*400*	*500*
1	45.43	54.48	61.02	64.71	67.48	71.84	75.61	78.12	78.71
2	51.47	59.26	66.05	69.15	72.59	79.04	79.04	81.89	82.48
3	54.90	61.78	67.56	70.83	74.27	80.72	80.72	83.65	84.16

Table 5-2: Evaluation of %ABD against document ranking.

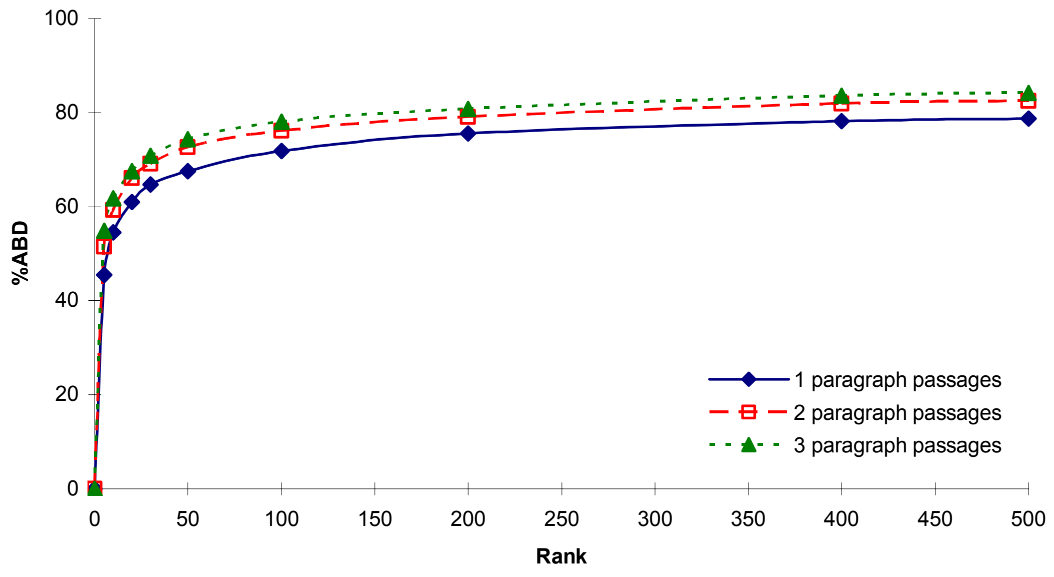


Figure 5-1: Evaluation of %ABD against document ranking.

Clearly the increase in %ABD does not start to level of towards a maximum until 100 documents have been retrieved, which is a cause for concern as the current system only uses the top twenty documents. Retrieving the top 500 documents still does not achieve the maximum %ABD of 90.02% suggesting that the IR engine is simply not able to locate some of the relevant documents.

Some groups, including Harabagiu et al. [Hara2000], have suggested using boolean retrieval methods instead of ranked methods (such as the vector space model), to guarantee that all the question terms are present in the returned documents. The main problem with boolean retrieval methods is that they do not provide a method of ranking the documents, hence you can not just decide which subset of those relevant documents you should use, unlike ranked retrieval where the approach is usually to use the top *n* relevant documents, as they should be more relevant than those below them in a ranked list. The approach to this problem taken in the paper by Harabagiu et al. is to have an iterative boolean retrieval loop. The question words are made into a query that is passed to the retrieval engine and an upper and lower bound is placed on the number of documents that may be returned from the retrieval step. If more or less documents are retrieved then the query is modified (see [Mold2000] for details as to how the query is modified) and the retrieval step is repeated until the number of returned documents is within the specified bounds.

Another possibility would be the combination of boolean and ranked retrieval. This may improve the IR performance by allowing only documents which contain the query words to be returned but at the same time applying a sorting to these documents. It is envisaged that this would work as a three step process:

1. Firstly a boolean retrieval would be carried out against the document collection, to retrieve just the relevant document identifiers.
2. Next a ranked retrieval would be carried out again to retrieve just the relevant document identifiers.
3. The intersection of the two sets of document identifiers is then taken as the list of relevant documents, keeping them in the order that they appeared in the ranked retrieval set.

Numerous issues arise from this process, such as what happens when there is no intersection between the two lists, or when the process does not return enough documents. At the moment it is unclear what the appropriate course of action should be in these situations Only experimentation will enable us to decide if this method is anymore successful than either of the two standard methods applied independently, and what should be done in the situations previously mentioned.

5.2 Are Natural Language Techniques Helpful?

Certainly some natural language techniques are essential, such as part-of-speech (POS) tagging, whereas full parsing, to produce syntax trees and semantics, may not be as useful as has been previously thought.

As an example consider answering all of the Who, Where and When questions from TREC 2001 simply by returning the most frequently occurring unique instance of the correct type (person, location, date) as tagged by a named entity (NE) transducer[25]. The results of using this simple approach and using our normal full NLP approach are detailed in Table 5-3, in which the best performing system, within each class of questions, is underlined.

Question Type	*System*	*MRR*	*Not Found (%)*
When	TREC 2002	0.394	14 (53.8%)
	Named Entity	0.396	11 (42.3%)
Where	TREC 2002	0.446	8 (30.8%)
	Named Entity	0.571	9 (34.6%)
Who	TREC 2002	0.324	28 (59.6%)
	Named Entity	0.298	27 (57.4%)
Combined	TREC 2002	0.375	50 (50.5%)
	Named Entity	0.395	47 (47.5%)

Table 5-3: Comparison of TREC 2002 system and NE frequency count.

In this simple study it is clear that nothing is gained from producing a full syntactic and semantic representation of the question and answer bearing documents (or at least not from the representations generated by our system). It is unclear how this simplistic approach could be extended to cope with other types of questions such as *"What is…"* as there is no single word that allows one to decide on the type of answer. If, however, there was a simple way to extend the system to cover all question types and the performance was comparable with that given in the above table then clearly the NLP

[25] Using the ANNIE NE Transducer included within the standard GATE 2 distribution.

processing used in our current system is completely redundant, as the named entity tagging is already taking place as part of that processing

5.3 Does Machine Learning Have a Role to Play?

Machine Learning (ML) techniques definitely have a role to play in modern question answering systems, be this the main method of answering questions or simply within an algorithm for sorting answers found using a NLP approach.

The best performing system at TREC 2001 relied solely on machine learning and contained no advanced NLP techniques (the only thing they used was tokenising and sentence splitting), see [Soub2001] for details. This surprised a lot of researchers as they had assumed that more involved processing of documents is required to be able to answer questions as reliably as this system appeared to do. This surprise was matched with interest and the system has been re-implemented by at least one group [Ravi2002].

Both these systems used a large corpus of questions and answers along with a text collection (usually the web) to generate a vast number of surface matching patterns. For example questions such as *"When was Mozart born?"* produce a list of patterns such as the following:

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

These patterns were obtained by taking the overlap between sentences known to contain the answer and query term (in this case Mozart). The patterns are then ranked by using the patterns to answer multiple question of the same type and seeing how accurate the patterns are, i.e. when a pattern is matched by the text does it match against a piece of text containing the correct answer or not. However simple this system appears it seems to work exceptionally well with the system from InsightSoft (see [Soub2001]) achieving an MRR score at TREC 2001 of 0.676, with the re-implemented system by Ravichandran and Hovy (see [Ravi2002]) claming similar results with an outstanding MRR of 0.86 on questions having a location as the answer.

5.4 Do Question Answering Systems Require World Knowledge

As with any AI task the question of whether or not world knowledge is required to answer questions needs to be addressed. Certainly many systems perform extremely well using very simple pattern matching techniques (see [Soub2001] and [Ravi2002]), but this could simply be a result of the class of questions asked at TREC (see section 2.2 for a discussion of question classes).

In [Deglin1996] Deglin and Kinsbourne discuss a number of experiments on patients undergoing electroconvulsive therapy (ECT) and specifically their ability to solve syllogisms (for a lay man's approach to this paper see chapter 8 of [McMan2002]). In the experiments most of the patients could correctly answer the syllogisms before treatment but their ability to answer them after therapy depended on which side of the brain had been treated (i.e. suppressed due to the ECT). Consider the following syllogism:

Every state has a flag. Zambia is a state. Does Zambia have a flag?

It is a straightforward question with no hidden trick and I am sure you all know that the answer is "*Yes, Zambia does have a flag*". Those patients who had received a right-sided shock (leaving the left half of the brain working) solved the problem in a very mechanical fashion such as:

It is written here that each state has a flag, and that Zambia is a state, therefore Zambia has a flag.

In other words they were strictly logical in their answers, whereas patients having received a left sided shock (leaving the right half of the brain working) gave answers similar to:

I've never been to Zambia and know nothing about its flag.

Clearly these patients were relying total on their world knowledge to answer the question and not approaching the question logically. This is very similar to the surface matching patterns used in [Soub2001] and [Ravi2002], they contain no world knowledge but are able to extract an answer purely from the text they are given. As we have already seen these systems perform exceptionally well in evaluations such as TREC, where the document collection is held as the universal truth, i.e. if a document contains an error which a system relies on to answer a question then the system is not penalised as the document is taken to be true even if it is not. As an example question 1396 (from TREC 2002) is *"What is the name of the volcano that destroyed the ancient city of Pompeii?"* which most people know to be *Vesuvius*. At least one document, however, misspells the volcano's name as *Vesuve*, and this is the answer returned by our system. Based on the TREC assumption that the document collection is always correct then this is a valid answer, even though it is clearly a misspelling of the mountain involved which with foreign names/places may be important and may not be obvious to the user of the system.

When confronted with a syllogism which contained a false premise such as:

"All monkeys climb trees. The porcupine is a monkey. Does the porcupine climb trees?"

the patients reacted very differently. Those who had received a right-sided shock (and were able to correctly answer the previous question) used the same approach to answer this question, namely:

Since the porcupine is a monkey it climbs trees.

Even when one of the experimenters pointed out that *"... you do know that a porcupine is not a monkey?"* the patients would reply along the lines of *"It is written on the card."* showing that they were solving the problem purely through the information they had been given. Patients who had been given a left sided shock (and had failed to correctly answer the previous question) usually responded with great indignation:

"Porcupine? How can it climb trees? It's not a monkey it's prickly like a hedgehog. Its wrong here!"

showing that again they were using their world knowledge this time being able to correctly point out the false hood in the question.

The conclusion of this experiment is that the brain contains not only the capacity for logical reasoning but also world knowledge, neither of these on their own is enough to correctly answer questions but combined they make a comprehensive system capable of answering different types of questions and ignoring incorrect information. The relevance of this experiment to question answering is that no matter how good systems are if they are relying solely on the documents from which they extract their answers then there is always a possibility of the answers being wrong due to a falsehood in the text. Only world knowledge will overcome this difficulty, although how it should be integrated into the current style of good performing QA systems is not clear.

6 Future Work

Numerous groups have recently reported great success in question answering using only simple surface matching patterns over a document collection. Most researchers have been stunned by the apparent success of these systems, and in one case even implemented the system themselves to prove that the process did function as well as had been reported. Clearly if these systems are performing better, or at least as well as, conventional NLP systems they should be studied in detail to see how the techniques can be improved or incorporated into those systems which are more NLP based.

One thing that is instantly clear is their simplicity and (usually) a complete lack of any advanced NLP techniques. I believe this is an area which needs more research. Consider for example what the benefits of adding techniques such as named entity tagging and anaphora resolution to these systems may be.

Named entity tagging could be used to restrict the type of answer allowed. For example, asking a question starting with the word *When* is clearly requesting some form of date or time, so combining the surface matching patterns with a named entity tagger to ascertain that the answer found (by the pattern) is indeed a date would surely remove some of the erroneous answers currently proposed by these systems.

It should be clear that these surface matching patterns are limited to only matching against text that has been seen during the training phase. This can be a problem, but one which it may be possible to reduce through the use of anaphora resolution. As an example one of the patterns usually mentioned for questions of the form *"When was X born?"* is:

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

where the pattern will match against text such as:

> *"Mark Greenwood (1979 -) Currently a student at the University of Sheffield".*

Clearly in this instance the proposed patterns will work without any problems and correctly extract the answer of *1979*. A pattern such as

```
<NAME> was born in <ANSWER>
```

can cause problems, however. Clearly it will match against texts such as:

> *"Mark Greenwood was born in 1979"*

but it would not be able to extract the correct answer from the text:

> *"Mark Greenwood is currently a student at the University of Sheffield. He was born in 1979".*

If, however, the text could be expanded through anaphora resolution of the pronouns in the text to give

> *"Mark Greenwood is currently a student at the University of Sheffield. Mark Greenwood was born in 1979"*

then the surface-matching pattern would be able to extract the correct answer, even though to a human the text no longer flows as well as the unexpanded version. The same argument can also be applied to matching name variations across a text. For example if the text used in the previous example was replaced by the following slight variation:

> *"Mark Greenwood is currently a student at the University of Sheffield. Mark was born in 1979".*

and the question text had been *"When was Mark Greenwood born?"* then the pattern still would not match against the text, however, a name matching algorithm would enable *Mark Greenwood* and *Mark* to be related and hence allow the text to be expanded in the same way as for the anaphora resolution.

Another possible expansion of the patterns would involve incorporating the inference rules as outlined in section 3.5.1. In this extension a pattern would include a place holder for text based on inference rules rather than just text taken straight from a document. For example the question *"Who wrote Macbeth?"* may generate, in the normal implementation, a pattern such as:

```
<ANSWER> wrote <NAME>
```

which would clearly match against the text

> *"Shakespeare wrote Macbeth"*

but would not match against

> *"Shakespeare was the author of Macbeth"*

as this would involve a second pattern, which may not have been generated if this form of construction had not been previously seen. The inference extraction technique would, however, be able to generate the inference:

> *"X wrote Y"* implies *"X was the author of Y"*

From the combination of inference rules and patterns could emerge a pattern of the form:

```
<ANSWER> <WROTE INFERENCE> <NAME>
```

where `<WROTE INFERENCE>` would match against *"wrote"*, *"was the author of"* or any other text which was in the same set of inferences. This would now allow the system to process both *"Shakespeare wrote Macbeth"* and *"Shakespeare was the author of Macbeth"* and extract *Shakespeare* as the correct answer. The downside to this (and any other) method of expanding the patterns is the increase in complexity and the possible loss of the patterns being easy to read and understand.

One thing these new breed of QA systems have in common is the use of some form of question classification, i.e. each question is answered using a specific set of patterns based on its type and the type of answer sought. These topologies are usually hand coded, although I believe that it may be possible to grow them from the questions and pattern sets.

Currently for each question type many questions are processed to produce the set of patterns for the question type. A possible alternative is to take each question and produce a set of patterns independently of all other questions. The questions can then be grouped into categories based on the overlap between the sets of patterns (i.e. if the pattern sets for two questions overlap by 100% then clearly they should be grouped together, although it would make sense to suggest that there is some level other than 100% at which two questions should be grouped together). This has not been investigated as part of this report and although the idea seems plausible problems such as stopping conditions etc would need to be investigated in detail.

The aim of my future study will be to determine the necessity and benefit of natural language techniques to the field of question answering. This will take the form of building a simple surface pattern matching system (similar to those in [Soub2001] and [Ravi2002]) which will hopefully incorporate an automatically generated question topology. The intention is to build the system as a set of GATE modules which should allow the system to be altered and expanded with relative by the addition of extra modules. These extra modules are likely to take the form of NLP techniques (such as named entity tagging and anaphora resolution). These modules can be evaluated as to their own independent performance (i.e. how well they do on their own task, for example NE tagging) and as to their impact on the performance of the end-to-end question answering process. Clearly with multiple modules comes the ability to evaluate the effectiveness of combining multiple techniques.

The result of this work will hopefully be not only an effective question answering system, but empirical details of how useful, effective or necessary numerous NLP techniques are to question answering. This may lead to some techniques being removed from consideration as useful, while promoting research into other techniques which show promise.

An extra benefit of this research should be a resource of questions and answer documents which will been tagged with information such as named entities and anaphora attachments etc. which will have been used to evaluate modules in this research but which could be used by other groups either as a resource for evaluating their own systems or as a resource for machine learning techniques.

7 Bibliography

[Anand2000] P. Anand, E. Breck, B. Brown, M. Light, G. Mann, E. Riloff, M. Rooth and M. Thelen. *Fun with Reading Comprehension.* Produced at Johns Hopkins Summer Workshop 2000. Available: http://www.clsp.jhu.edu/ws2000/groups/reading/index.shtml (2nd September 2002)

[Ask2002] *What is Ask Jeeves?* [online]. Available: http://www.ask.co.uk/docs/about/what_is.asp (13th August 2002).

[Bamf2001] J. Bamford. *Building an Interface for an On-Line Question Answering System.* Unpublished undergraduate dissertation, The University of Sheffield, 2001.

[Bobr1977] D. Bobrow, R. Kaplan, M. Kay, D. Norman, H. Thomson and T. Winograd. *GUS, a Frame Driven Dialog System.* Artificial Intelligence, 8:155-173, 1977.

[Brill1992] E. Brill. *A Simple Rule-Based Part-of-Speech Tagger.* Proceedings of the Third Conference on Applied Natural Language Processing, pages 152-155, Trento, Italy, 1992.

[Brill2001] E. Brill, J. Lin, M. Banko, S. Dumais and A. Ng. *Data-Intensive Question Answering.* Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

[Buch2001] S. Bucholz and W. Daelemans. *Complex Answers: A Case Study using a WWW Question Answering System*. Journal of Natural Language Engineering, Vol. 7, No. 4 (2001).

[Burg2000] J. Burger, C. Cardie, V. Chaudhri, R. Gaizauskas, S. Harabagiu, D. Israel, C. Jacquemin, C. Y. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, E. Voorhees and R. Weishedel. *Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A)*. October 2000. Available: http://www-nlpir.nist.gov/projects/duc/roadmapping.html.

[Burt1997] N. Burton-Roberts. *Analysing Sentences: An Introduction to English Syntax (2nd Edition).* Longman, 1997.

[Coop2000] R. J. Cooper and S. M. Rüger. *A Simple Question Answering System.* The Ninth Text REtrieval Conference (TREC 9), 2000.

[Cope1993] J. Copeland. *Artificial Intelligence: A Philosphical Introduction.* Blackwell Publishers Ltd, 1993.

[Cope1990] A. Copestake and K. Sparck Jones. *Natural Language Interfaces to Databases.* The Knowledge Engineering Review, 5(4):225-249, 1990.

[Cunn2002a] H. Cunningham. *GATE, a General Architecture for Text Engineering*. Computers and the Humanities, volume 36, pp. 223-254, 2002.

[Cunn2002b] H. Cunningham, D. Maynard, K. Bontcheva and V. Tablan. *GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications*. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia, July 2002.

[Deglin1996] V. L. Deglin and M. Kinsbourne. *Divergent Thinking Styles of the Hemispheres: How Syllogisms Are Solved during Transitory Hemisphere Suppression.* Brain and Cognition, 31, pages 285-307 (196).

[Gaiz1996] R. Gaizauskas, H. Cunningham, Y. Wilks, P. Rodgers and K. Humphreys. *GATE – An Environment to Support Research and Development in Natural Language Engineering.* In Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-96), pages 58-66, Toulouse, France, October 1996.

[Gee1983] R. Gee and C. Watson. *The Usborne Book of Better English.* Usborne Publishing, 1983.

[Green1961] B. F. Green, A. K. Wolf, C. Chomsky and K. Laughery. *BASEBALL: An Automatic Question Answerer.* In Proceedings of the Western Joint Computer Conference 19, pages 219-224 (1961). Reprinted in [Grosz1986], pages 545-549.

[Grosz1986] B. J. Grosz, K. Sparck Jones and B. L. Webber, editors. *Readings in Natural Language Processing.* Morgan Kaufmann, Los Altos, CA (1986).

[Hara2000] S. Harabagiu, D. Moldovan, M. Paşca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Gîrju, V.Rus and P. Morărescu. *FALCON: Boosting Knowledge for Answer Engines.* The Ninth Text REtrieval Conference (TREC 9), 2000.

[Hara2001] S. Harabagiu, D. Moldovan, M. Paşca, M. Surdeanu, R. Mihalcea, R. Gîrju, V.Rus, F. Lăcătuşu, P. Morărescu and R.Bunescu. *Answering complex, list and context questions with LCC's Question-Answering Server.* Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

[Haug1985] J. Haugland. *Artificial Intelligence: the Very Idea.* MIT Press, 1985.

[Heiser1980] J. F. Heiser, K. M. Colby, W. S. Faught and R. C. Parkinson. *Can Psychiatrists Distinguish a Computer Simulation of Paranoia from the Real Thing?* Journal of Psychiatric Research, 15, pages 149-162, 1980.

[Hepple2000] M. Hepple. *Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based POS Taggers.* Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000), pp278-285, Hong Kong, October 2000.

[Hirs1999] L. Hirschman, M. Light, E. Breck and J. Burger. *Deep Read: A Reading Comprehension System.* In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, 1999.

[Hirs2001] L. Hirschman and R. Gaizauskas. *Natural Language Question Answering: The View From Here.* Journal of Natural Language Engineering, Vol. 7, No. 4 (2001).

[Hirst1998] G. Hirst and D. St-Onge. *Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms.* In Fellbaum 1998, pp. 305-332.

[Hovy2001] E. Hovy, L. Geber, U. Hermjakob, C. Lin and D. Ravichandran. *Towards Semantics-Based Answer Pinpointing.* Proceedings of the DARPA Human Language Technology conference (HLT). San Diego, CA, 2001.

[Hump1998] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham and Y. Wilks. *Description of the LaSIE-II System as Used for MUC-7.* Proceedings of the Seventh Message Understanding Conference (MUC-7), 1998.

[Hump1999] K. Humphreys, R. Gaizauskas, M. Hepple and M. Sanderson. *University of Sheffield TREC 8 Q & A System.* The Eighth Text REtrieval Conference (TREC 8), 1999.

[Jarv1993] G. Jarvie. *Bloomsbury Grammar Guide.* Bloomsburry Publishing, 1993.

[Jiang1997] L. Jiang and D. Conrath. *Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy.* In proceedings of International Conference on Research in Computational Linguistics, Taiwan, 1997.

[Kwok2001] C. Kwok, O. Etzioni and D. Weld. *Scaling Question Answering to the Web.* ACM Transactions in Information Systems, Vol 19, No. 3, July 2001, pages 242-262.

[Lea1998] C. Leacock and M. Chodorow. *Combining Local Context and WordNet Similarity for Word Sense Identification.* In C. Fellbaum, editor, WordNet: An Electronic Lexical Database, chapter 11, pages 265-284. MIT Press, 1998.

[Lehn1977] W. Lehnert. *A Conceptual Theory of Question Answering.* Proceedings of the Fifth International Joint Conference on Artificial Intelligence, pages 158-164, 1977. Reprinted in [Grosz1986] pages 651-657).

[Light2001] M. Light, G. Mann, E. Riloff and E. Breck. *Analyses for Elucidating Current Question Answering Technology.* Journal of Natural Language Engineering, Vol. 7, No. 4 (2001).

[Lin1998] D. Lin. *An Information-Theoretic Definition of Similarity.* In Proceedings of the 15th International Conference on Machine Learning, Madison, WI, 1998.

[Lin2001] D. Lin and P. Pantel. *Discovery of Inference Rules for Question Answering.* Journal of Natural Language Engineering, Vol. 7, No. 4 (2001).

[McMan2002] C. McManus. *Right Hand, Left Hand: The Origins of Asymmetry in Brains, Bodies, Atoms and Cultures.* Orion Publishing Group, 2002.

[Mill1995] G. A. Miller. *WordNet: A Lexical Database.* Communication of the ACM, vol 38: No11, pages 39-41, November 1995.

[Mold1999] D. Moldovan, S Harabagiu, M. Paşca, R. Mihalcea, R. Goodrum, Roxana Gîrju and V. Rus. *LASSO – A Tool for Surfiing the Answer Net.* The Eighth Text RErieval Conference (TREC 8), 1999.

[Mold2000] D. Moldovan, S. Harabagiu, M. Paşca, R. Mihalcea, R. Goodrum, R. Gîrju and V. Rus. *The Structure and Performance of an Open-Domain Question Answering System.* Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000), pages 563-570, 2000.

[Mold2002] D. Moldovan, M. Paşca, S. Harabagiu and M. Surdeanu. *Performance Issues and Error Analysis in an Open-Domain Question Answering System.* In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pages 33-40, Pennsylvania, 2002.

[Oh2001] J.H. Oh, K.S. Lee, D.S. Chang, C. Seo and K.S. Choi. *TREC-10 Experiments at KAIST: Batch Filtering and Question Answering.* Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

[Plam2001] L. Plamondon, G. Lapalme and L. Kosseim. *The QUANTUM Question Answering System*. Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

[Poli2000] J. Polifroni and S. Seneff. *Galaxy-II as an Architecture for Spoken Dialogue Evaluation*. Proceedings of the Second International Conference on Language Resources and Evaluation (LREC), Athens, Greece, May 31st - June 2nd, 2000.

[Ravi2002] D. Ravichandran and E. Hovy. *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, 2002.

[Resnik1995] P. Resnik. *Using Information Content to Evaluate Semantic Similarity.* In Proceedings of the 14th International Joint Conference on Artificial Intelligence, pages 448-453, Montreal, 1995.

[Rilo2000] E. Riloff, and M. Thelen. *A Rule-based Question Answering System for Reading Comprehension Tests*. ANLP/NAACL-2000 Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems

[Rob2002] I. Roberts. *Information Retrieval for Question Answering.* MsC Dissertation, The University of Sheffield, UK, 2002.

[Schank1977] R. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding*. Hillsdale, 1977.

[Scott2000] S. Scott and R. Gaizauskas. *University of Sheffield TREC 9 Q & A System.* The Ninth Text REtrieval Conference (TREC 9), 2000.

[Sene1998] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue. *Galaxy-II: A Reference Architecture for Conversational System Development*. Proceedings of ICSLP 98, Sydney, Australia, November 1998.

[Shaw1991] M. Shaw, J. Wood, R. Wood, H Tibbo. *The Cystic Fibrosis Database: Content and Research Opportunities*. LISR 13, pp. 347-366, 1991

[Simm1965] R. F. Simmons. *Answering English Questions by Computer: A Survey.* Communications of the ACM, 8(1):53-70 (1965).

[Soub2001] M. M. Soubbotin. *Patterns of Potential Answer Expressions as Clues to the Right Answers.* Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

[Turing1950] A. M. Turing. *Computing Machinery and Intelligence.* Mind 59, pages 433-460, 1950.

[Voorh1999] E. Voorhees. *The TREC 8 Question Answering Track Report.* The Eighth Text REtrieval Conference (TREC 8), 1999.

[Voorh2001] E. Voorhees. *Overview of the TREC 2001 Question Answering Track*. Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

[Weiz1966] J. Weizenbaum. *ELIZA – A Computer Program for the Study of Natural Language Communication Between Man and Machine.* Communications of the ACM, 9, pages 36-45, 1966.

[Wino1972] T. Winograd. *Understanding Natural Language.* Academic Press, New York, 1972.

[Woods1973] W. Woods. *Progress in Natural Language Understanding – An Application to Lunar Geology.* In AFIPS Conference Proceedings, volume 42, pages 441-450 (1973).

[Zue2000] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen and L. Heatherington. *JUPITER: A Telephone-Based Conversational Interface for Weather Information.* IEEE Transactions on Speech and Audio Processing, 8(1):100-112, 2000.

Appendix 1: Index of Figures

Figure 3-1: Original screen display of SHRDLU in action..11
Figure 3-2: An example reading comprehension test..15
Figure 3-3: Performance of systems over TRECs 8, 9 and 10. 20
Figure 4-1: Original system setup. 23
Figure 4-2: Original QA-LaSIE system modules. 23
Figure 4-3: The layout of the QA system in GATE 2..27
Figure 4-4: The hypernym trees for *fish* and *food*...29
Figure 4-5: The shortest path from *fish* to *food*...30
Figure 4-6: An application for assisting the development of grammars. 36
Figure 4-7: The interface to our new WWW QA system...39
Figure 4-8: SHAPAQA User Interface...40
Figure 5-1: Evaluation of %ABD against document ranking. 43

Appendix 2: Index of Equations

Equation 2-1: Equation for calculating MRR. 6
Equation 2-2: The TREC 2002 scoring measure.......... 7
Equation 4-1: Equation for scoring an answer instance. 26
Equation 4-2: The Leacock and Chodorow Semantic Similarity equation. 29
Equation 4-3: Equation used to calculate semantic similarity in our system. 30
Equation 4-4: Equation, used in TREC 2002, for scoring an answer instance.......... 31

Appendix 3: Index of Tables

Table 2-1: Comparison of Casual Questioner and Professional Information Analyst. 2
Table 2-2: Class distribution of TREC questions.. 3
Table 3-1: MRR scores of systems over TRECs 8, 9 and 10. 20
Table 4-1: Table showing distances and scores for the three paths.. 30
Table 4-2: Results of using Google to boast system score. 34
Table 4-3: Results of combining score and number of occurrences... 38
Table 5-1: Results of IR experiments and their effects on the QA system. 42
Table 5-2: Evaluation of %ABD against document ranking.. 43
Table 5-3: Comparison of TREC 2002 system and NE frequency count. 44