The University of Sheffield TREC 2002 Q&A System

Mark A. Greenwood, Ian Roberts and Robert Gaizauskas
{m.greenwood,i.roberts,r.gaizauskas}@dcs.shef.ac.uk

Department of Computer Science
University of Sheffield
Regent Court, Portobello Road
Sheffield S1 4DP UK

1 Introduction

The system entered by the University of Sheffield in the question answering track of TREC 2002 represents a significant development over the Sheffield system entered into TREC-8 [9] and TREC-9 [15], although the underlying architecture remains the same. The essence of the approach is to pass the question to an information retrieval (IR) system which uses it as a query to do passage retrieval against the text collection. The top ranked passages output from the IR system are then passed to a modified information extraction (IE) system. Syntactic and semantic analysis of these passages, along with the question, is carried out to identify the *"sought entity"* from the question and to score potential matches for this sought entity in each of the retrieved passages. The potential matches are then combined or discarded based on a number of criteria. The highest scoring match is then proposed as the answer to the question.

2 System Description

2.1 Overview

The key features of the question answering system, for processing a single question, are shown in Figure 1. Firstly the TREC document collection is indexed using the probabilistic Okapi information retrieval system (this is done once only in advance of any questions) [14]. This index is then used to return the top n passages relevant to the question, the query to Okapi being the question words. The top n passages are then submitted along with the question to QA-LaSIE, our modified IE system, which should produce one or more answers.

The reasoning behind this architecture is straightforward. The text collection is too large to be processed in its entirety by the IE system. It is, however, the IE system which is capable of carrying out the detailed linguistic analysis needed to answer the questions. IR systems, however, are specifically designed to process huge amounts of text, and to return the result of a query in a short space of time. Using an IR system as a filter between the text collection and the IE system should allow us to benefit from the systems' respective strengths.

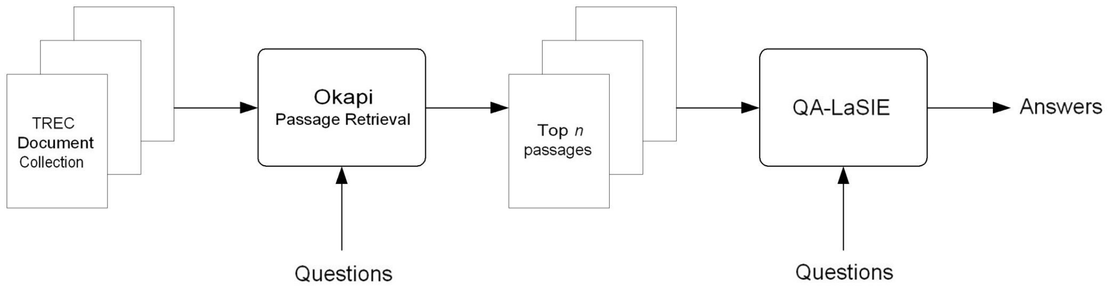

Figure 1: System setup for the question answering task.

Passage Length	***Coverage***	***Correct Answers (out of 100)***
1 paragraph	67%	13
2 paragraphs	74%	11
3 paragraphs	72%	7
4 paragraphs	72%	8
5 paragraphs	70%	7
6 paragraphs	70%	7
7 paragraphs	71%	7
full documents	72%	7

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

2.2 Okapi

An important question involving the information retrieval component of the QA system is how much text to return. We must decide a) how many documents to retrieve and b) how much retrieved text per document to return (passage size).

For TREC 2002 we decided to process the top 20 documents returned by Okapi. Experimentation not complete at the time of the test run subsequently showed we should have considered about five times this number of documents as on average the top twenty documents only contained answers for about 60% of the questions while the top 100 documents contained on average answers for about 85% of the questions [13] (of course the more documents examined per question, the greater the number of entities which can potentially be confused with the answer).

Okapi supports passage retrieval which can be parameterised by setting a minimum passage length, a maximum passage length and a step value controlling how the passage window is moved over the text. We experimented with different sizes of passage using a random sample of 100 TREC-9 and TREC 2001 questions as queries against the TREC-2001 document collection. These experiments are documented in [13] and their main results are detailed in Table 1. The definition of the data in each column of the table is as follows:

Coverage the percentage of questions for which at least one relevant answer bearing passage was found in the retrieved data.

Correct Answers the number of questions for which the exact answer returned by the system matched one of the Perl patterns supplied for that question.

TREC Score the TREC 2002 confidence score for the run.

From these results it seems that using passages of one paragraph in length gives the best performance (13 of the 100 questions were answered correctly), even though the best coverage is provided for passages of length two paragraphs. As the main experiments were not completed in time, two of the submitted runs used the retrieval techniques previously used with our question answering system (i.e. passages of up to three paragrpahs in length, see [15]) and one run used passages of just one paragraph to attempt to confirm the experimental results.

2.3 LaSIE

The basis of the question answering system is the LaSIE information extraction system, originally developed to participate in the Message Understanding Conference evaluations [8]. LaSIE operates inside the GATE platform [4], and as a new version of GATE has become available [3] since our participation in TREC-9, LaSIE has been ported to use the new version, leading to a few minor changes.

The system is essentially a pipeline of modules each of which process the entire text before the next module is invoked. The following is a brief description of each of the modules in the LaSIE system:

Tokeniser Identifies token boundaries and text section boundaries.

Gazetteer Identifies single and multi-word matches against multiple domain specific full name and keyword lists, and tags matching phrases with appropriate name categories.

Sentence Splitter Identifies sentence boundaries in the text body.

POS Tagger A rule-based part-of-speech tagger [6].

Tagged Morph Simple morphological analysis to identify the root form and inflectional suffix for tokens that have been tagged as noun or verb.

NE Transducer Identifies names of people, organisations etc.

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 quasi-logical form (QLF) of each sentence is constructed.

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.

2.4 QA-LaSIE

The QA-LaSIE system takes as input a question and a set of passages retrieved by the IR system and outputs the highest ranked answer. When multiple questions are processed by the system it outputs one answer per question ranking the answers based on how confident it is in the answers.

Figure 2 shows the end-to-end layout of the system as entered in TREC 2002. Four key alterations were made to the original LaSIE IE system for entry into the question answering track at TREC-8 and TREC-9 and these have been developed further for this year's entry. These alterations are as follows:

1. the grammar used by the parser was extended to cover question types;

2. the discourse interpreter was modified to allow the QLF representations of each question to be matched against the discourse model of a candidate answer text;

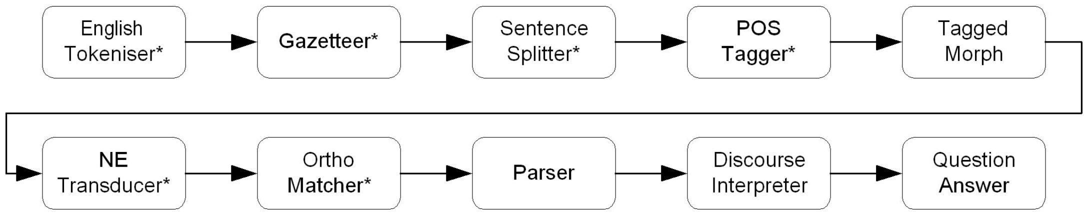

Figure 2: QA-LaSIE system modules (* denotes a standard GATE 2 module).

3. an answer identification procedure which scored all discourse entities in each candidate text as potential answers was added to the discourse interpreter;
4. a Question Answer module was added to examine the discourse entity scores across all passages, determine the ranking of the answers and then output the appropriate answer text.

Exact details of these changes would not sufficiently explain the essence of the approach taken to question answering by the QA-LaSIE system. Therefore the following sections describe the key processes involved in our approach to question answering.

2.4.1 Parsing: Syntactic and Semantic Analysis

Questions were one of the sentence constructions not handled by the original LaSIE parser. Extra grammar rules were developed to cover the example questions that were available. The syntactic grammar rules have a semantic component that is used to build a QLF representation of the question. 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 `e`*N*. The use of the word *Who* in the question suggests the answer will be a person and so `person(e1)` is added to the QLF. Also the `qvar` is set to `e1` showing that the question is seeking a person (as `person` and `qvar` share the same entity). The relational predicates `lsubj` (logical subject) and `lobj` (logical object) link any verb arguments found in 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 and the entity identifiers are replaced by question entity identifiers of the form `q`*N* (i.e. `e1` becomes `q1`, `e2` becomes `q2` etc.) to facilitate later processing.

Candidate answer texts are processed in exactly the same fashion although the grammar rules do not instantiate a `qvar` and the entity identifiers are not altered.

2.4.2 Resolution of Question and Candidate Answer Texts

After a candidate answer text has been parsed the QLFs are passed to the discourse interpreter. This behaves 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 or discourse model of all the entities and relationships present in the multiple QLFs for a document. This is built by running a coreference algorithm against the semantic representation of successive sentences as they become available, in order to unify them with the discourse model built so far. This results in multiple references to the same entity across the text being merged into a single unified instance.

Given this discourse model of a text, the QLF of the question is added to the model as the first sentence and coreference is then carried out between question entities (qN) and entities within the text (eN).

The method for determining and scoring each candidate answer is then as follows:

1. Each sentence in a candidate answer document is given a constraint score, C, equal to 1 point for each question constraint that matches a member of the sentence, where a question constraint is a unary predicate specifying the type of an entity (eg. person is the type of `person(eY)` in the question.

2. Within each sentence every remaining entity (`eY`) is tested for:

   (a) Semantic Similarity to the `qvar`, S: the reciprocal of the length of the path between the type of the `qvar` entity and the type of `eY` in the semantic lattice (ontology) or if this fails (usually because the two entities are not both present in the system's small ontology) the reciprocal of the Leacock-Chodorow distance [10] between the `qvar` and `eY` in WordNet [12]. For instance if `qvar` and `eY` are of the same type then they will receive a score of 1.

   (b) Object Relation, O: 0.25 if `eY` is related to a question constraint within the sentence by apposition, a qualifying relationship, or with the prepositions *of* or *in*.

   (c) 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 and `eY` stands in the same relation to an event entity of the same type as `qvar` does.

These three values are then combined with the scores for the sentence and the number of question constraints, Q, to give Equation 1 (where 2.8 is a normalising factor determined via experimentation).

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

The discourse interpreter then returns all the candidate answers and their associated scores for processing by the answer module.

2.4.3 Answer Output and Ranking

Due partly to the differences between TREC 2002 and the previous question answering tracks and partly to the move to using the new version of GATE, the final Question Answering module has been completely redeveloped and a number of new ideas have been included, which are outlined in this section.

The limitations of window-based methods for pinpointing answers have been discussed in numerous papers including [7]. The main concerns with these methods are:

- It is impossible to accurately pinpoint the boundaries of an answer (e.g. an exact name or phrase).

- These rely solely on word level information and do not use semantic information (hence no knowledge of the type, such as person or location, of the answer being sought).

- It is impossible to see how such methods could be extended to composing an answer from many different documents or even from different sentences or phrases within a single document.

One way to filter out some inappropriate candidate answers is to assume 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 exact 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 candidate answer strings before we even look at the score assigned to them. Word overlap between a question and candidate answer can be expressed 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 therefore be discarded (an exception is TREC 2001 question 1026 *"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 between 0% and 100% overlap it is unclear whether the candidate answer may or may not be correct. Our system simply discards any candidate answers which overlap 100% with the question they seek to answer.

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 answers which we are allowed to return. On some occasions we were actually returning identical answers (i.e. for Q1000 *"The sun's core, what is the temperature?"* 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.

Furthermore it may be possible to prune candidate answers that are substrings of longer candidate answers, when the question is suitably vague; as is the case in the question *"Where is Perth?"* to which our system returns a list of ranked answers containing: *Australia* and *Western Australia.* Clearly *Australia* and *Western Australia* are both acceptable answers to the question, so only one of them need be returned.

The approach taken to deal with these answer strings, similar to that used in [1], is to test if two proposed answers A and B are the similar 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 a possible answer.

Applying the same approach to the question *"In which country is Perth?"* would not be as effective, since *Western Australia* is not an exact country name so while this method is better than simple string matching approaches, there is still scope for 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 peoples names: *'Armstrong'* becomes *'Neil A. Armstrong'* and *'Davis'* becomes *'Eric Davis'*, etc.

These techniques (overlap, similar answers, etc.) are then used to discard, merge and rank the candidate answers found within the document collection for a single question (full details of the ranking algorithm can be found in [5]).

The method of ranking single answers to multiple questions (i.e. to produce the confidence sorted list required for this years submission) is based on the following attributes of each answer:

- the score (the higher the better)
- the number of other answers which were semantically the same as this one (the higher the better)

- the IR system rank of the document from which the answer originates (the lower the better as the top document returned by the IR step is ranked 1)

2.4.4 Answering Questions Requiring Multiple Answers

List questions are inherently harder to answer than standard, single answer questions, mainly because systems have to combine information from multiple sources to locate the required number of answers. Also a 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:

1. The system processes the question in the usual way, producing a long list of ranked answers.
2. The question is then scanned, token by token, until the first token whose part-of-speech signifies that it is a number. This is then assumed to be the number of answers sought.
3. The requested number of answers is then returned from the top of the ranked list.

Clearly this approach suffers from the problem that some questions may contain more than one number, i.e. *"In the 2001 US Presidential election who were the 2 main candidates?"*. This problem did not surface during the evaluation, however, as our system correctly identified the number of answers to return for all the questions.

2.4.5 Boosting Performance using Answer Redundancy

As has been reported in Light et al. [11], the number of answer instances to a given question in the document collection (within a single document or multiple documents each containing the answer once) directly effects 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 the answers are likely to occur in different contexts, giving the parser a better chance of analysing at least one of them 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 constructed from the snippets displayed on the Google results page for the top ten documents returned by Google. These snippets are certainly not full documents, and are rarely full sentences but this is not a problem as the bottom-up chart parser we employ is not constrained to only selecting full sentence or complex phrase categories. This method of using just the snippets has been shown to be successful in [2], 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, 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 (other answers found using Google are simply discarded).

Over a sample of one hundred questions (TREC questions 1000 to 1099) the results of combining the collections in this way (based on returning the top five answers for each question) can be seen in Table 2.

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

Table 2: Results of using Google to boost system score.

		Not			*Confidence*	*No Answer*	
Run Tag	*Wrong*	*Supported*	*Inexact*	*Right*	*Score*	*Precision*	*Recall*
`sheft11mo3`	422	9	18	51	0.128	0.162	0.130
`sheft11mog3`	394	12	22	72	0.203	0.150	0.130
`sheft11mog1`	389	11	20	80	0.222	0.150	0.065

Table 3: Results from the three main track entries.

3 Results and Analysis

3.1 Results Observed During Development

Unfortunately we did not enter the system into TREC 2001 and so there were no official scores for the system over that question set. However, the first development task was to produce unofficial scores for our unaltered TREC-9 system over the TREC 2001 questions, using the regular expression patterns kindly made available by NIST. The result was that the unaltered TREC-9 system achieved a mean reciprocal rank (MRR) score of 0.169, over the TREC 2001 questions compared to its official TREC 9 MRR score of 0.206 (both using answers of 50 bytes or less). This drop in performance may be due to the fact that our system is not designed to handle definition style questions, which made up a significant percentage of the TREC 2001 question set. This system now scores an MRR of 0.343 over the TREC 2001 question set, clearly a significant improvement over the previous system.

3.2 Final Evaluation Results

3.2.1 Main Track

We submitted three runs to the main question answering track. The differences between the runs all concernes the size and composition of the document collection generated for a single question, these were:

`sheft11mo3`: This run used the top twenty passages retrieved from the AQUAINT collection by Okapi. The maximum length of a passage was three paragraphs, the minimum was one paragraph.

`sheft11mog3`: This run used the same collection as did `sheft11mo3`, augmented with the top ten snippets returned by Google when given the question as a search query.

`sheft11mog1`: This run is the same as `sheft11mog3` except the passages retrieved by Okapi from the AQUAINT collection are limited to at most one paragraph in length.

Table 3 shows the full evaluation results for the three different runs over the 500 test questions. From this it can be seen that the sheft11mog1 run was the best of the three configurations, suggesting that using documents from more than one source is beneficial, and also that documents of one pargraph in length are more suited to this work than longer documents, confirming what was demonstrated during development (see Section 2.2).

3.2.2 List Track

We submitted two runs to the list track. The differences between the runs concern the size and composition of the document collection generated for a single question, these were:

`sheft11lo`: This run used the top twenty passages retrieved from the AQUAINT collection by Okapi. The maximum length of a passage was a single paragraph.

`sheft11log`: This run used the same collection as `sheft11lo`, augmented with the top ten snippets returned by Google when given with the question as a search query.

Unfortunately we did not have time to test the list answering system with the result that the system scored an average accuracy of only 0.06 (for both runs). A serious flaw in the processing of list questions was subsequently discovered, although fixing this resulted in a system whose average accuracy was only 0.09.

4 Conclusions and Future Work

At its core, Sheffield's entry in this year's QA track remains the same as our TREC-9 system in 2000 [15]. There were, however, a number of enhancements, the most significant were:

- using a semantic similarity metric over WordNet as one factor in determining the score of candidate answer entities;
- filtering the final ranked answer list
  - to remove duplicate and near-duplicate answers and simultaneously boost the remaining candidate's rank;
  - to eliminate answers which completely overlap with the question;
- employing Google to search the Web for documents relevant to a given question and boosting the rank of answers found by the QA system both in the Google-returned document snippets and the TREC collection.

Each of these enhancements produced small but noticeable improvements. Ideas for future work include:

- expanding the size of the document set passed on from the IR system to the QA system – experiments not completed till after the TREC run showed that for 40% of the questions in a test sample the QA system was simply not receiving any document containing an answer;
- experimenting with adaptive algorithms to optimise the weightings of the various features used to rank the answer candidates.

References

[1] Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. Data-Intensive Question Answering. In *Proceedings of the Tenth Text REtrieval Conference*, 2001.

[2] S. Bucholz and W. Daelemans. Complex Answers: A Case Study using a WWW Question Answering System. *Natural Language Engineering*, 7(4), 2001.

[3] H. Cunningham. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254, 2002.

[4] 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.

[5] Mark A. Greenwood. Question Answering. First year PhD Progress Report, Department of Computer Science, The University of Sheffield, UK. Available, October 2002, from http://www.dcs.shef.ac.uk/˜mark/phd/work/index.html, 2002.

[6] Mark Hepple. Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based POS Taggers. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, pages 278–285, Hong Kong, October 2000.

[7] Eduard Hovy, Laurie Geber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. Towards Semantics-Based Answer Pinpointing. In *Proceedings of the DARPA Human Language Technology Conference (HLT)*, San Diego, CA, 2001.

[8] 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. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[9] Kevin Humphreys, Robert Gaizauskas, Mark Hepple, and Mark Sanderson. University of Sheffield TREC-8 Q & A System. In *Proceedings of the 8th Text REtrieval Conference*, 1999.

[10] 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 265–285. MIT Press, 1998.

[11] Marc Light, Gideon S. Mann, Ellen Riloff, and Eric Breck. Analysis for Elucidating Current Question Answering Technology. *Natural Language Engineering*, 7(4), 2001.

[12] George A. Miller. WordNet: A Lexical Database. *Communications of the ACM*, 38(11):39–41, November 1995.

[13] Ian Roberts. Information Retrieval for Question Answering. MSc Dissertation, Department of Computer Science, The University of Sheffield, UK. Available, Februray 2003, from http://www.dcs.shef.ac.uk/teaching/eproj/msc2002/abs/m1ir.htm, 2002.

[14] S. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In *Proceedings of the 8th Text REtrieval Conference*, 1999.

[15] Sam Scott and Robert Gaizauskas. University of Sheffield TREC-9 Q & A System. In *Proceedings of the 9th Text REtrieval Conference*, 2000.