The University of Sheffield's TREC 2004 Q&A Experiments

Robert Gaizauskas, Mark A. Greenwood, Mark Hepple, Ian Roberts, Horacio Saggion
r.gaizauskas,m.greenwood,
m.hepple,i.roberts,saggion } @dcs.shef.ac.uk
Department of Computer Science
University of Sheffield, UK

1 Introduction

The experiments detailed in this paper are a continuation of the experiments started as part of the work undertaken in preparation for participation in the TREC 2003 QA evaluations as documented in Gaizauskas et al. [2003]. Our main experiments for TREC 2004 were concerned with investigating: a) alternative approaches to information retrieval (IR) for question answering b) alternative approaches to answer extraction for list and factoid questions, and c) alternative approaches answering definitional or 'other' questions. In each of these three areas we have developed two principal alternatives, each of which has variants. Given the TREC limit of three test runs per site, we have not been able to evaluate properly all combinations of these approaches. Consequently, the systems we submitted only give a partial picture of work carried out, and further evaluation is underway. In the following we describe the major alternatives we have been exploring in these three areas and present the formal test results for the system combinations we submitted.

2 Information Retrieval for Question Anwering

Like many others, we approach the QA task by first carrying out a retrieval step whose goal is to retrieve documents likely to contain answers to factoid, list or nugget-requesting questions. Analysis of previous year's TREC results had convinced us that poor performance in this first step was having a serious negative impact on overall QA performance.

Attracted by the relative transparency of boolean approaches to IR and the positive remarks made about their utility for QA by other TREC participants [Harabagiu et al., 2000], for TREC 2003 we implemented a boolean search engine called MadCow which does word indexing at the sentence level. This year we decided to explore systematically strategies for boolean retrieval for the purpose of QA measuring retrieval effectiveness in terms of *coverage* and *answer redundancy* [Roberts and Gaizauskas, 2004]. As a control we used Lucene, a freely available open-source IR engine that supports a boolean query language while performing ranked retrieval using the standard `tf.idf` weighting scheme with the cosine similarity measure[1].

2.1 MadCow

A number of experiments were carried out aiming at reinforcing our understanding of query formulation, search and post-hoc ranking for question answering. We have explored the following questions (see Saggion et al. [2004] for details):

- *question analysis*: is it worth doing linguistic analysis of the question? if so, what forms of analysis are useful?
- *term expansion*: which strategies should be pursued? is synonymy expansion or morphological variant expansion helpful?
- *query broadening*: are measures of a term's discriminative power of use when broadening the search query? if so, is it better to prefer terms of high or low discrimination when selecting terms to discard?
- *passage and match window size*: is there any benefit in matching boolean query terms across windows that are larger in size than a single sentence? is there any benefit in returning additional text around the window in which matching is constrained to occur?
- *ranking*: how should one rank sentences returned in a boolean environment, so that the best possible sentences are given first to the answer extraction component?

After exploring various strategies we settled on the following combined question analysis, search and post-hoc ranking strategy. The question is analysed and three sets of terms are identified: proper names, quoted expressions, and other terms. Our search strategy treats these three groups differently during query formulation and post-hoc ranking of results. The search is iterative and stops when the target number of sentences has been returned or when there is no further possible reformulation. The search begins with the conjunctive query created from all question terms, and then iteratively drops terms until at least one matching sentence is returned. This deletion process first drops common terms, then name terms,

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

Document Rank	*MadCow*		*Lucene*	
	coverage	*redundancy*	*coverage*	*redundancy*
1	12.17	0.121	12.17	0.121
5	26.08	0.408	32.17	0.465
10	34.34	0.617	39.56	0.686
20	42.60	0.917	47.39	0.978
30	45.21	1.108	51.30	1.204
50	50.00	1.443	55.65	1.465

Table 1: Comparison of coverage and redundancy (factoid questions only)

and then finally quote terms. Within each of these groups terms are dropped in order of increasing informativeness (i.e. decreasing document frequency order). Terms are morphologically expanded before the query is issued. At each iteration sentences not retrieved in the previous iterations are ranked by their degree of similarity to the original question. The post-hoc ranking method used differentially weights question terms Q in the quote, name and common groups, as in (1). For the experiments reported here, we stipulated a weight value $w(t)$ of 1/6 for common terms, 2/6 for name terms, and 3/6 for quote terms.

$$score(P,Q) = \sum_{t \in Q \cap P} w(t) * idf(t) \quad (1)$$

The matching process was constrained so that query terms had to match within and single sentence; and that one sentence passage was returned.

2.2 Lucene

In our experiments with Lucene the AQUAINT collection is split into passages (using the paragraph markers in the source documents) and the resulting paragraphs have stopwords removed and are stemmed using the Porter stemmer Porter [1980] before being indexed. Average paragraph length is about 1.5 sentences.

Passages are retrieved using queries consisting of all the question words (Lucene subsequently removes stopwords and stems remaining terms before accessing the index). The ranking algorithm prefers short passages containing all the question words over longer passages and those which do not contain all the question words.

Discussion Comparative results of MadCow versus Lucene are shown in Table 1. From the table we see that for all ranks beyond 5 Lucene is scoring about 5 percentage points higher then MadCow on coverage (percentage of questions for which a text returned at or below the given rank contains an answer). Two observations are worth making. First, while MadCow coverage is lower, it returns less text (one sentence on average as opposed to 1.5 sentences for Lucene). Since at least of our answer extraction components fares worse with more text (see results of the two MadCow based runs in Section 5). Secondly, the performance of both IR systems is inadequate. Coverage performance bounds answer extraction performance and as a consequence answer extraction components using either of these search engines at rank 50 will be unable to answer morethan about 50% of the questions regardless of how good they are.

3 Approaches to Answer Extraction for Factoid and List Questions

For the TREC 2003 QA evaluation we introduced a baseline QA system against which we intended to compare our approach to question answering embodied in the QA-LaSIE system which had been developed over several TREC cycles [Gaizauskas et al., 2003]. The official TREC judgments, however, showed that our baseline approach actually outperformed QA-LaSIE when answering factoid questions and was also quite capable of answering list questions, performing only slightly worse than QA-LaSIE. This motivated us to put considerable energy into refining that baseline system this year, described below in Section 3.2. However, some alteration and refinements were made to the QA-LaSIE system which we retained as our other answer extraction component.

3.1 Matching on Logical Forms

QA-LaSIE performs partial syntactic and semantic analysis of questions and candidate answer bearing documents and then performs matching over the derived logical form representation. The system has been described in detail in past TREC proceedings (see, e.g. Greenwood et al. [2002]) and here we will only describe modifications.

Question Processing The format of the questions for this year complicates matters for us, because the QA-LaSIE answer extraction system relies on the target being a part of the question. To deal with this we adopted the naive expedient of transforming the factoid and list questions associated with each target by replacing pronominal expressions by the target. Examples of transformation can be seen in table 2. While this works for cases such as the first example, it does not for cases such as the second, where there is a definite NP that needs replacing by the target rather than a pronoun. As a consequence, in such cases the identify of the target is missing for document retrieval and answer extraction.

The transformation process guesses whether the target is a 'term' or a 'person' by relying on the distribution of pronouns in associated target questions (if the distribution of singular personal pronouns in the target questions is greater than the

Target	*Original Question*	*Transformed Question*
James Dean	How did he die?	How did James Dean die?
Crips	What does the name mean or come from?	What does the name mean or come from?

Table 2: Question transformation

distribution of other pronouns in the target questions, then the target is classified as a person otherwise it is classified as a common term). This was done in order to inform the process of nugget extraction for 'other' questions which discriminates between 'person' targets ('who' questions in TREC 2003) and common term targets ('what' questions in TREC 2003).

Factoid and List questions Only very limited changes were made to the QA-LaSIE system used in the TREC 2003 QA evaluation [Gaizauskas et al., 2003]. Changes include updated named entity detection and improved question parsing, allowing a substantially larger set of questions to parsed. One significant change concerns the handling of list questions. Instead of returning all answers found for a list question, the system now returns only the top five answers in an attempt to increase the precision of the answer sets.

3.2 The Shallow Multi-Strategy Approach

This year the system that had served as a baseline in our TREC 2003 submission has been improved in order to answer a wider variety of questions (one of its major downfalls in 2003).

While this approach to question answering is simpler than the QA-LaSIE approach it would now be wrong to describe it as a simple or baseline approach to answer extraction. Since it relies on a variety of shallow approaches which are tried in sequence, a descriptively accurate, if somewhat cumbersome designation is "a shallow multi-strategy approach", abbreviated to SMS in the following.

Before describing how the SMS system answers the different question types we describe aspects common to its handling of all question types. Unlike the QA-LaSIE system the SMS system does not attempt to rephrase the questions to contain the target (see Section 3.1). Rather, the target is simply appended to the question if it is not already present. This results in slightly different questions being processed by this system, illustrated by Table 3, which shows how the questions in Table 2 are modified under this approach. This approach has the advantage that it ensures that the target is present both when performing IR and answer extraction. Also it does not rely on examining all questions for a given target to determine whether the target is a term or person, so as to know which pronouns to replace.

Target	*Original Question*	*Transformed Question*
James Dean	How did he die?	How did he die? James Dean
Crips	What does the name mean or come from?	What does the name mean or come from? Crips

Table 3: Question transformation

Factoid Questions The answering of factoid questions is actually accomplished via three different approaches to question answering working in a conditional pipeline (i.e. the system only moves onto using the next approach if it fails to find an answer to the question using the current approach). The three approaches are surface matching text patterns, semantic type extraction and WordNet lookup.

1. *Surface Matching Text Patterns* The approach to QA using surface matching text patterns is as documented in Greenwood and Saggion [2004] (without the fallback to semantic entities) and includes pattern sets to answer questions as listed in Table 4 which also shows the accuracy of these sets measured against documents on the web.

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

Table 4: Evaluation of the individual pattern sets.

   This approach was actually part of the `shef12simple` approach in the TREC 2003 evaluation although it was not mentioned in Gaizauskas et al. [2003] as none of the patterns actually produced answers to any of the TREC 2003 questions for which they were suitable.

2. *Semantic Type Extraction* The semantic type extraction system is a development of the system detailed in Greenwood [2004]. The main development here is an improved answer ranking function.

First, the semantic type of the expected answer is determined from a set of 60 known types using a set of hand-coded rules which operate over the question text. The question is then submitted to Lucene to retrieve the twenty most relevant passages from AQUAINT. All entities of the expected answer type (not including any within the question) are then extracted from the documents. The extracted entities are then grouped together using the equivalence test [Brill et al., 2001]: *two answers are said to be equivalent if all of the non-stopwords in one answer are present in the other answer or vice versa*. When two entities are deemed equal then the one which was extracted from the sentence with the greatest word overlap with the question is retained on the assumption that the more words in common between the question and a candidate answer-bearing sentence, the more likely that sentence is to contain a correct answer to the question. Along with each entity grouping is stored the frequency of occurrence within the relevant documents as well as the sentence from which the most likely member was extracted.

The extracted and grouped entities are then ranked using a scoring function, given that a unique answer a to question q has been seen C_a times by the answer extraction component within retrieved documents the most likely of which occurred in sentence s then the answer is scored using the equation:

$$score(a, q, s) = C_a * \frac{|q \cap s|}{|q|} \tag{2}$$

This scoring function takes into account, in the same way the grouping algorithm did, that it is more likely that a correct answer will not only appear frequently in relevant documents but will also come from a sentence which contains many (if not all) the question words.

3. *WordNet Lookup* The main problem with the semantic type extraction approach to question answering is simply the number of questions which are assigned an UNKNOWN type, i.e. questions for which the system has no idea what the expected answer type should be and so does not attempt to answer. Clearly the answer type hierarchy could be extended to cover new answer types as detection systems are developed for new semantic categories. As an intermediate strategy, however, it is possible to modify dynamically the answer type hierarchy using WordNet to increase the questions which can be successfully classified and possibly answered.

   This addition to the system works by attempting to determine the semantic type of the word or phrase in a question which specifies the thing being sought. For example Q1940 *"What grapes are used in making wine?"* is clearly looking for the names of grape varieties. Now assuming the system can correctly extract *grape* from the question after determining that is the thing begin asked for then the system can insert grape as a top level entry in answer type hierarchy. We stated earlier that the hierarchy is constructed to cover only those types which the answer extraction component can locate in free text. Given this the system needs a way to locate grape varieties in free text. The system simply uses WordNet to construct a list of all known grape varieties by extracting the hyponyms the grape entries. This list is then used to tag all the grape varieties in the texts being considered. The answers are then extracted, grouped and ranked in the same way as in the semantic type extraction system.

List Questions When answering list questions the SMS system uses both the semantic type extraction and WordNet lookup approaches as used for answering factoid questions returning multiple answers to each question.

For the TREC 2003 evaluation the system simply returned all the entities of the correct type which were extracted as answers to the list questions. Unfortunately whilst this approach was capable of finding correct answers to the list questions it also retrieved many wrong answers to each question, swamping the correct answers and resulting in relatively high recall but exceptionally low precision.

For the TREC 2004 evaluation the SMS system adopts a more sensible approach to limit the number of answers returned per question without reducing the number of correct answers returned. If the system finds 10 answers or less for a question then they are all returned. On the other hand if more than 10 answers are located then the first 10 are returned along with any further answers which have a confidence score (see Equation 2) of 0.08 or above. The cutoff point of 0.08 was determined after examination of the answers returned by the system to the TREC 2003 list questions and was the point at which the system returned the fewest answers to each question (increasing the precision) without discarding a single correct answer (i.e. the recall was not affected).

4 Approaches to Nugget Identification and Extraction

Our approach to answering definition questions in TREC 2003 worked quite well, achieving scores that placed us in the top 10 at this subtask. However, we were also surprised by the excellent performance of the TREC 2003 definition baseline system submitted by Xu et al. [2003] Consequently we decided not only to continue development of our earlier approach but also to investigate the simple approach as well, augmenting it with a number of shallow methods for reducing the amount of non-informative text included in its output.

4.1 The Target-Enrichment + Filter Approach

The approach we proposed for definition questions in TREC 2003 essentially involved three steps [Gaizauskas et al., 2003]: (i) web-based knowledge acquisition for each target – terms are gathered to supplement a query typically consisting of a single target term (e.g. *battery* in *What is a battery*) in order to constrain the set of documents returned to include those likely to contain definitional nuggets (ii) document retrieval for each target, and (iii) nugget identification and filtering from the returned set of documents. Since steps (i) and (iii) are the interesting ones from the definition question point-of-view it seems reasonable to dub this a "target-enrichment + filter approach". This basic strategy has been retained for TREC 2004 and most of the work on this approach this year went into the nugget identification process.

During the web-based knowledge acquisition phase (see Saggion and Gaizauskas [2004]) relevant terms associated with each nugget are identified. Term acquisition is done by extracting terms that co-occur with the target in definition bearing sentences found on Web pages, Wikipedia pages, and Britannica pages. The result of the process is a list of terms Rel(TARGET) for each TARGET. Terms t have an associated weight $weight(t)$ which is the number of times t and TARGET co-occur in the sources examined. This year, we expanded the process so that the list of mined terms would include all morphological variants and nominalisations of terms co-occurring with the target, overcoming some limitations identified in our 2003 definition system. Each term is recorded with its associated frequency of occurrence. Examples of terms identified for targets 'Horus' and 'Crips' are shown in Table 5.

Target	***Co-occurring Terms***
Horus	falcon-headed, god, solar, Egyptian, deity, ...
Crips	graffi ti, art, gangs, gang, los, angeles, ...

Table 5: Example Expansion Terms

The nugget identification step looks for evidence to classify a sentence as 'definition bearing'. The following sources of evidence are used: presence of the target or target alias in the sentence; presence of relevant terms found on external sources; and presence of definition patterns. Two types of patterns were used this year: lexical patterns ("X which is", "like X", "X and others", etc.) and part-of-speech and named entity patterns.

Exclusion terms The restriction that a nugget for a target should not contain information already contained in an answer to a factoid or list question for that target was addressed in the following way. For each TARGET a list of terms (with *idf* weights), called $exclusion$(TARGET), was created from the relevant sentences used to answer factoid and list questions for the target. The list is used during nugget identification.

Inducing POS definition patterns Because the lexical definition patterns we used last year were far too constrained to match naturally occurring definitions, this year we decided to induce definition patterns using last year's definition data. The induced patterns have two forms: (i) TARGET X_2 X_3 X_4, or (ii) Y_1 Y_2 Y_3 TARGET, where TARGET matches the question target or an alias and X_i and Y_i are: part-of-speech tags, dates, or titles. Each pattern has a weight which represents its relative importance. Patterns are induced and used separately depending on the type of target (person or other entity). For example a pattern for a person could be "TARGET , WD VBD" that matches a sentence like "...Aaron Copland, who composed...". The weighted definition patterns were induced in the following way:

- for each target and nugget from last year's TREC/QA definition data, a collection of sentences from AQUAINT was constructed. The question target and the text nugget were used as a query submitted to Lucene and the top 10 sentences were retrieved (ranks 1 to 10). For example for the question "Who is Aaron Copland" one of the queries submitted was "Aaron Copland american composer".
- each sentence retrieved by Lucene was automatically marked with the target, POS information (NNP, VB, etc.), dates, and titles (GATE tools were used for analysis – see `http://gate.ac.uk`).
- a coreference algorithm, provided in GATE, was run to identify references to the target and those coreferencing expressions were marked as TARGET as well.
- sequences of four elements TARGET X_2 X_3 X_4 and Y_1 Y_2 Y_3 TARGET were collected, the score associated to the sequence was $1/sentence_rank$. So patterns found in the most 'relevant' sentences get score 1 and those found in the least relevant sentence get score 0.1.
- scores for each pattern were summed for each instance
- patterns were translated into JAPE grammar rules for use in a GATE pattern recogniser.

Nugget identification and filtering Each of the sentences retrieved from the collection is analysed and the following scores computed:

- main entity score ($Main(S)$): which is 1 if the sentence contains the target (i.e., Franz Kafka), 0.5 if the sentence contains a target alias (i.e., Kafka), and 0 otherwise;
- related terms score: $REL_Terms(S) = \sum_{T \in (rel(TARGET) \cap S)} weight(T)$

Target	***Definition Bearing Sentences***
Crisps	to the FlyPlaya Web site will see the words 'C' z Up,"a greeting used by the Crips, an infamous street gang
Horus	Osiris, the god of the underworld, his wife, Isis, the goddess of fertility, and their son, Horus, were worshiped by ancient Egyptians.

Table 6: Nuggets

Run Tag	*Factoid*	*List*	*Other*	*Combined*
ShefMadCow20	0.061	0.058	0.321	0.125
ShefMadCow50	0.043	0.053	0.317	0.114
shef04avf	0.213	0.125	0.312	0.216

Table 7: Summary results from our three main task entries.

- definition pattern score ($DEF_Pattern(S)$): which is 1 if the sentence matches a definition pattern and 0 otherwise
- POS pattern score: $POS_Patterns(S) = \sum_{P \text{ matches } S} weight(P)$
- exclude score: $Exclude_Terms(S) = \sum_{T \in (exclusion(TARGET) \cap S)} weight(T)$

Each sentence is sorted in descending order by (in order) $Main(S)$, $REL_Terms(S)$, $DEF_Patterns(S)$, $POS_Patterns(S)$, and in ascending order by $Exclude_Terms(S)$. So, no sentence returned by the search engine is in principle rejected. However this sorting is expected to rank relevant nuggets higher than irrelevant ones. Nuggets are output in rank order until a maximum of 4000 characters is reached. Nuggets are not included in the answer set if they are regarded as too similar to a previously output nugget. Two nuggets are considered too similar if they have 50% or more of their tokens in common.

4.2 The Bare Target + Filter + Reduce Approach

Our initial attempt at developing a baseline system for answering definition (or 'other') questions was a slightly simplified implementation of the TREC 2003 baseline system developed by Xu et al. [2003]. Sentences were selected only if they contained the target as it appears in the question; no use of coreference was made to increase the number of matching sentences. Each sentence was retained if it did not overlap more than 70% with any sentence already in the definition. The process stopped either when there were no more sentences to process or the definition had reached 4000 characters in length. This approach while effective still results in much repetition within the resulting definitions. For instance one of the nuggets for Q2274 *"Who is Alice Rivlin?"* is that she was the Federal Reserve Vice Chair. The baseline approach constructed a definition which contained 18 sentences containing only this nugget – clearly this will have a dramatic effect on the performance of the system.

In an attempt to remove more of the redundant sentences from the definitions a second filtering step was introduced. This second filter works by calculating the sum of the percentage overlap of increasingly longer n-grams. The n-grams considered range from length 1 (a single token) to length s which is the length of the shortest of the two sentences being compared. From limited testing a cutoff level of 50 was determined with pairs of sentences having a score above this being deemed equivalent. This had the desired effect reducing the number of sentences containing the nugget Federal Reserve Vice Chair for Q2274 from 18 to 7. To increase the number of nuggets returned, rather than reduce the amount of text, the system was updated to create initially a definition of up to 5000 characters. The second filter is then applied and the resulting definition is trimmed to the first x sentences that produce a definition of 4000 characters. A conservative evaluation of this system shows an F(β=5) score of 0.5199.

While filtering the sentences allows the system to remove some of the redundancy from the generated queries, it is clear that returning whole sentences still results in a large amount of redundant text being included in the definition. Rather than attempting to extract the salient details from the sentences we attempted to determine a number of phrases and clauses which while being redundant could also be removed from the sentence without affecting either the meaning or flow of the text. To allow the system to answer questions in real-time we only attempt to find redundant words or phrases using shallow methods which do not require intensive processing. This rules out detection of redundant phrases which would require full syntactic or semantic parsing to identify. A number of sources were consulted for possible ways in which redundant phrases could be both identified and safely removed [Dunlavy et al., 2003, Purdue University Online Writing Lab, 2004]. The phrases were removed from the sentences before any filtering is applied as previous work in summarization has shown this to be the most effective point at which to remove redundant clauses [Conroy et al., 2004]. This updated system when evaluated gives an F($\beta = 5$) score of 0.5398, a slight improvement over the system without redundant phrase removal.

The words and phrases which were deemed redundant and easily removable were: imperative sentences, gerund clauses, leading adverbs, sentence initial expletives, redundant category labels, unnecessary determiners and modifiers, circumlocutions, unnecessary that and which clauses, and noun forms of verbs.

5 Final Evaluation Results

From the development and experimentation detailed above, we configured three evaluation runs as follows.

ShefMadCow20 This run consisted of using MadCow to retrieve relevant documents using the strategy described in Section 2.1 and then using QA-LaSIE to extract answers for factoid and list questions. The number of documents used for each question was restricted to 20. To answer 'other' questions, we used the target enrichment + filter system described in Section 4.1. 1000 top documents retrieved by MadCow using the target as a query were used in the process. question we could have answered using 20 sentences is 42% of the questions.

ShefMadCow50 This run is similar to ShefMadCow20. For factoid and list questions 50 documents returned by

MadCow were used. For 'other' questions, 1000 top passages retrieved by Lucene using the question target as a search term were used and nuggets were extracted using the target enrichment + filter method.

shef04afv This run answers questions using the shallow multi-strategy approach to answer extraction for factoid and list questions detailed in Section 3.2. Lucene was used as the IR system and the top 20 passages were passed on to the SMS answer extractor. This run used the bare target + filter + reduce approach to answering 'other' questions. One major issue with this run is that answers to the list and factoid questions are not removed from the answers to the 'other' questions as each question is answered independently which may have led to a drop in performance of the 'other' questions compared to evaluations during development.

The results from these three runs can be seen in Table 7.

6 Discussion

Much more analysis of results needs to be carried out before firm conclusions can be drawn. However, certain observations are worth making even now:

- The IR components are not performing adequately and more research needs to be carried out to determine how to boost their performance.
- The QA-LaSIE answer extraction component performs worse when given 50 top sentences as opposed to 20, even though more questions have answers within the top 50 sentences than within the top 20. Hence its selectivity needs improvement.
- The shallow multi-strategy approach scores much more highly than the QA-LaSIE approach, especially on factoid questions. However, it is important to keep in mind that they were coupled to different retrieval engines. Comparison over identical sets of retrieved documents needs to be undertaken to understand more fully their differences.
- Both approaches to 'other' questions performed reasonably well and it is not immediately clear where they differ and the difference is significant.

References

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

John M. Conroy, Judith D. Schlesinger, John Goldstein, and Dianne P. O'Leary. Left-Brain/Right-Brain Multi-Document Summarization. In *Proceedings of the Document Understanding Conference (DUC 2004)*, 2004.

Daniel M. Dunlavy, John M. Conroy, Judith D. Schlesinger, Sarah A. Goodman, Mary Ellen Okurowski, Dianne P. O'Leary, and Hans van Halteren. Performance of a Three-Stage System for Multi-Document Summarization. In *Proceedings of the Document Understanding Conference (DUC 2003)*, 2003.

Robert Gaizauskas, Mark A. Greenwood, Mark Hepple, Ian Roberts, Horacio Saggion, and Matthew Sargaison. The University of Sheffield's TREC 2003 Q&A Experiments. In *Proceedings of the 12th Text REtrieval Conference*, 2003.

Mark A. Greenwood. AnswerFinder: Question Answering from your Desktop. In *Proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics (CLUK '04)*, pages 75–80, Birmingham, UK, January 7 2004.

Mark A. Greenwood, Ian Roberts, and Robert Gaizauskas. The University of Sheffield TREC 2002 Q&A System. In *Proceedings of the 11th Text REtrieval Conference*, 2002.

Mark A. Greenwood and Horacio Saggion. A Pattern Based Approach to Answering Factoid, List and Definition Questions. In *Proceedings of the 7th RIAO Conference (RIAO 2004)*, Avignon, France, April 27 2004.

Sanda Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morărescu. FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of the 9th Text REtrieval Conference*, 2000.

Martin Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

Purdue University Online Writing Lab. Conciseness: Methods of Eliminating Wordiness. `http://owl.english.purdue.edu/handouts/print/general/gl_concise.html` [July 2004], 2004.

Ian Roberts and Robert Gaizauskas. Evaluating Passage Retrieval Approaches for Question Answering. In *Proceedings of 26th European Conference on Information Retrieval*, 2004.

Horacio Saggion and Robert Gaizauskas. Mining on-line sources for definition knowledge. In *Proceedings of FLAIRS 2004*, Florida, USA, 2004. AAAI.

Horacio Saggion, Robert Gaizauskas, Mark Hepple, Ian Roberts, and Mark A. Greenwood. Exploring the Performance of Boolean Retrieval Strategies for Open Domain Question Answering. In *Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering (IR4QA)*, Sheffield, UK, July 29 2004.

Jinxi Xu, Ana Licuanan, and Ralph Weischedel. TREC2003 QA at BBN: Answering Definitional Questions. In *Proceedings of the 12th Text REtrieval Conference*, 2003.