Answer Selection in a Multi-Stream Open Domain Question Answering System

Valentin Jijkoun and Maarten de Rijke

Language & Inference Technology Group, University of Amsterdam
Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands
E-mail: jijkoun, mdr@science.uva.nl

Abstract. Question answering systems aim to meet users' information needs by returning exact answers in response to a question. Traditional open domain question answering systems are built around a single pipeline architecture. In an attempt to exploit multiple resources as well as multiple answering strategies, systems based on a multi-stream architecture have recently been introduced. Such systems face the challenging problem of having to select a single answer from pools of answers obtained using essentially different techniques. We report on experiments aimed at understanding and evaluating the effect of different options for answer selection in a multi-stream question answering system. We examine the impact of local tiling techniques, assignments of weights to streams based on past performance and/or question type, as well redundancy-based ideas. Our main finding is that redundancy-based ideas in combination with naively learned stream weights conditioned on question type work best, and improve significantly over a number of baselines.

1 Introduction

Question answering is a variation on the traditional document retrieval task where, in response to a user's question, an answer has to be returned instead of a ranked list of relevant documents from which the user has to extract an answer herself.

Traditional question answering systems typically consist of a single processing stream that performs three steps in a sequential fashion: question analysis, search, and answer selection [10, 14, 16, 17]. These systems typically focus on the corpus from which the answers are to be extracted, and they may use a small number of additional resources, either unstructured (such as the Web), semi-structured (such as WordNet or the CIA World Fact Book), or structured (such as geography databases). Essentially, such single-stream systems adopt a "one size fits all" approach, treating factoid questions of all types in the same manner.

Recently, a number of teams have adopted more complex architectures for their question answering systems, either involving feedback loops as part of a single stream architecture [15], or involving multiple streams that somehow implement different answering strategies [1, 2, 4, 11]. The motivations underlying

these multi-stream approaches are two-fold: 1. Some answering strategies may be highly effective for certain question types, but not for others. 2. An important practical benefit of multi-stream architectures is easy modification, maintenance, and testing of the different subsystems as well as easy integration of multiple source of information.

One of the big challenges raised by multi-stream approaches to open domain question answering is that at some stage a global choice needs to be made: which of the many candidate answers produced by the independent streams is to be chosen as *the answer* to be returned by the system? The important thing to notice here is that techniques and information sources used by different streams can be very different: some can be more reliable than others and the scoring methods of the streams may be incompatible. Think, for instance, of a Web answering stream based on redundancy and a knowledge base lookup stream. In such a combined system, the task of reducing answer candidates "to a common denominator" is highly non-trivial.

In this paper we experiment with a number of answer selection strategies in a multi-stream question answering environment. We generalize and develop the methods described earlier [1, 4], present novel techniques for combining answer streams and systematically evaluate the performance of different methods in different combinations, using our own multi-stream question answering system for a case study. One of our main findings is that, using a fairly limited amount of data, a relatively simple machine learning method performs just as well as humans in assigning weights to streams, measured in terms of the performance of the overall system.

The selection methods we consider do not make any assumptions about the precise nature of the streams involved, or even about the answer types being used in question classification. All we need is data on the past performance of the streams. For these reasons, we believe that the methods and results that we present below are widely applicable in the question answering domain.

In the remainder of this paper, we first discuss related work on answer selection and result merging. We then describe the architecture of one particular multi-stream question answering architecture. Next, we discuss in detail different aspects of answer selection in such architectures. Finally, we report on experiments with different answer selection schemes and discuss our findings.

2 Related Work

In a single-stream question answering environment, the goal of answer selection is to choose from a pool of answer candidates the most likely answer for the question. There are many approaches to answer selection in the single-stream setting. Here, we only have space to mention a few. In their TREC 2002 system, the BBN team used the standard single-stream pipeline by using a document retrieval system to select documents that are likely to contain answers to a given question and then ranking candidate answers based on the answer contexts using the same retrieval system [20]. They then used a few constraints to re-rank

the candidates; these constraints include whether a numerical answer quantifies the correct noun, whether the answer is of the correct location sub-type and whether the answer satisfies the verb arguments of the question. Like BBN's question answering system, the system developed by LCC is an example of a single-stream architecture whose answer selection process is very knowledge-intensive [15]. It incorporates lots of AI-like technology, by actually attempting to prove candidate answers from text, with a number of feedback loops and sanity-checking that can reject answers or require additional checking. By way of contrast to these knowledge-intensive selection and reranking methods in single-stream architectures we mention a purely data-driven approach due to Magnini et al. [13]. They employ the redundancy of the Web to re-rank answer candidates found in the collection, by using hit counts for question and answer terms.

Let's turn to multi-stream question answering environments now. Here, the task of selecting the final answer is complicated by the fact that the final answer has to be selected from several pools of ranked candidates found by different streams. We need to compare answers coming from different, often incomparable sources and pick the one that is most likely to be correct.

IBM's PIQUANT question answering system used at TREC 2002 [1] implements a multi-strategy and multi-source approach. It faces a problem of combining answers coming from external resources (such as the Web) and answers extracted from the reference corpus. The question answering evaluation methodology of TREC requires that all answers be justified in the corpus, so the proposed solution is to use answer feedback: for each external answer candidate the best (if any) relevant passage in the reference corpus is found and used for candidate ranking. Because of this answer feedback, all answering streams produce ranked candidate answers in a uniform fashion, which obviously simplifies the final answer selection.

Some multi-stream systems solve the answer selection problem based solely on the stream that produced it. E.g., the University of Waterloo's MultiText system consults multiple resources, using a variety of methods, ranging from shallow parsing to word n-gram mining [2]. In MultiText, these approaches are organized in two streams, one called the early-answering subsystem (which consults ready-made databases), the other called statistical answer-selection (which resembles the traditional single-stream question answering pipeline). Final answer selection is based purely on the stream that delivered the answer: whenever the early-answering stream produces an answer, MultiText judges it to be correct, based on the fact that whenever this stream does produce an answer, it is usually correct.

Another possibility that has been explored in the literature is to favor answers that are returned by the highest number of streams. At TREC 2002, the LIMSI team experimented with an architecture consisting of two streams, one based on the local text collection and one based on the Web [4]. The underlying assumption is that similar answers coming from very different sources are more reliable, so the answer selection module favors those answer candidates found in the local collection, which have also been found in Web documents.

In the experiments on which we report below, we compare answer selection strategies that incorporate and build on ideas from both the MultiText and LIMSI approaches. Before reporting on those experiments, we briefly point to related work, not in question answering but in document retrieval, where the combination of retrieval runs is one of the recurring themes. It goes back at least to [7], and a recent overview of combination approaches is given in [3], describing combinations of document representation, query formulations, ranking algorithms, and search systems. The standard combination methods for merging ranked document lists are discussed and compared in [7, p.245/246].[1]

- *combMAX* Take the maximal query-document similarity score of the individual runs.
- *combMIN* Take the minimal similarity score of the individual runs.
- *combSUM* Take the sum of the similarity scores of the individual runs.
- *combANZ* Take the sum of the similarity scores of the individual runs, and divide by the number of non-zero entries.
- *combMNZ* Take the sum of the similarity scores of the individual runs, and multiply by the number of non-zero entries.
- *combMED* Take the median similarity score of the individual runs.

Since the similarity scores produced by different systems may differ radically, one often finds that a score normalization step takes place before one of the above methods is applied. In the question answering setting evaluations are not based on mean average precision (MAP) scores, but, in effect, on p@1. Because of this, there is no point in considering methods that may hurt early precision (but benefit MAP), such as combMIN, combANZ, and combMED. In our experiments in Section 5 we do consider answer selection methods based on the same intuitions as combMAX, combSUM, and, especially, combMNZ.

3 Overview of the Quartz Question Answering System

To make matters concrete, and to prepare for a report on our answer selection experiments, we will now zoom in on our own multi-stream open domain question answering, called Quartz. Quartz exists in two incarnation, a Dutch language version [12], and an English language version [11]. The brief overview below is based on the English version; we refer to [11] for more details.

After analyzing an incoming question, Quartz sends it to six streams in parallel, each of which is a small question answering system on its own. Every stream produces a ranked list of answer candidates (an answer pool). At the end of the process, the six answer pools are merged and the best candidate is returned as the final answer. While they share some components (named entity and part-of-speech taggers, parser, lexical resources, retrieval engine), the streams are independent and use very different strategies for answer extraction. The streams making up Quartz are briefly described as follows:

[1] In the setting of multi-lingual document retrieval, *round robin* is another merging method sometimes used [9]. Since we have to return a single final answer in the question answering setting, this method is not relevant for our discussions.

Table Lookup. This stream uses specialized knowledge bases constructed by preprocessing the collection. The stream exploits the fact that certain types of information (such as country capitals, abbreviations, and names of political leaders) tend to occur in a small number of fixed patterns. Similar to [6] we developed a small number of patterns for offline extraction of this information, using surface text and syntactic templates and WordNet. The knowledge base currently consists of 15 specialized tables. A fairly intricate knowledge base lookup process allows for non-exact matches.

Pattern Matching. Inspired by the success of methods based on pattern matching for certain question type [18], this stream exploits the fact that in some cases, the contextual format of an answer to a question can be back-generated from the question itself. For example, an answer to a question such as *2257. What is the richest country in the world?* may match the pattern `<Capitalized-Words>(,| is) the richest country in the world`. For each incoming question a set of possible answer patterns is generated and matches are attempted in relevant documents. We have two streams implementing this approach, *Web Patterns* and *Collection Patterns*, that use Google and our in-house IR engine, respectively, to retrieve relevant documents from the Web or from the AQUAINT corpus.

Ngram Mining. This stream, similar in spirit to [5], constructs a weighted list of queries for each question using a shallow reformulation process and then looks at word ngrams in the relevant retrieved document snippets. Quartz uses two variations of this stream: *Web Ngrams* and *Collection Ngrams*, using the Web and the local AQUAINT corpus, respectively.

Tequesta. Tequesta is a stream that implements a linguistically informed approach to QA. We refer to [11] for more details.

Each of the six streams described above provides a confidence score for each answer candidate. However, the actual values of the scores are calculated in a stream-specific way. The *Table Lookup*'s confidence depends on the number of occurrences of the relevant fact in the database and on the "exactness" of the match. The candidates' scores in the *Collection Patterns* and *Web Patterns* streams are based on the manually assigned accuracy of the patterns and the frequencies of the found answers. The *Web Ngrams* and *Collection Ngrams* streams use the number of ngram occurrences and some other features (e.g., presence of named entities of the appropriate type). Finally, the confidence scores of *Tequesta* stem from document scores given by the retrieval engine, the distance between question and answer terms in documents as well as a few additional syntactic, semantic and statistical features. Taking into account these differences, our answer selection module brings the candidates from all six streams together and selects the final answer.

Since Quartz's streams employ essentially different answering techniques, we expected that different streams would perform well on different question types. Indeed, an analysis of the results of the overall system and of the individual

streams shows that each stream finds correct answers that are not found by other streams. Table 1 presents an evaluation of our best run at the TREC 2003 question answering track; we only consider factoid questions here, leaving out so-called list question and definition questions as these were assessed and scored differently. Evaluation is done by us, using the answer patterns provided by NIST [19].[2] We show the performance of our six streams for all questions and for questions of the 4 most frequent question types: *location* (e.g. *2316. What is the largest city in Austria?*), *number-many* (e.g. *1979. How many moons does Venus have?*), *date* (e.g. *1924. When was the first hair dryer made?*) and *person-ident* (e.g. *2301. What composer wrote "Die Götterdämmerung"?*). The row *alone* gives the number of questions correctly answered by the stream alone, the row *increase* gives the number of questions that the system would not answer without the stream (i.e., the number of questions correctly answered by the full system, but not answered by the system with this stream disabled).

Table 1. Comparison of the performance of the six question answering streams implemented in Quartz, on all question types and on the four most frequent question types.

		Correct answers						
Questions		*Table Lookup*	*Collection Patterns*	*Web Patterns*	*Collection Ngrams*	*Web Ngrams*	*Tequesta*	Total #q's
All	alone	71	39	51	39	65	63	413
	increase	17	1	6	3	30	30	
location	alone	16	9	2	9	17	15	67
	increase	2	1	1	3	10	10	
number-many	alone	5	5	5	5	10	17	46
	increase	0	0	0	0	2	9	
date	alone	8	3	8	3	7	14	37
	increase	1	0	2	0	4	3	
person-ident	alone	6	5	6	6	10	2	31
	increase	0	0	1	0	3	1	

The numbers in Table 1 clearly indicate that all of Quartz' six streams contribute to the system's performance, but the significance of the contribution depends on the question type. While each stream does find answer candidates

[2] Observe that these patterns do not distinguish between so-called *correct*, *inexact* and *unsupported* answers.

not found by any of the others (see the rows labeled *increase* in the table), many answers are found simultaneously by several streams, which explains the difference between *increase* and *alone* values.

4 Answer Selection in Quartz

In a single-stream question answering environment, the goal of answer selection is to choose from a pool of answer candidates the most likely answer for the question. As we pointed out in our discussion of related work, in a multi-stream environment, the task is complicated by the fact that the final answer now has to be selected from several pools of ranked candidates found by different streams. We need to compare answers coming from different sources and pick the one that is most likely to be correct. In this section we describe the way this has been implemented in Quartz.

4.1 Reranking and Score Normalization using Web

As described before, all of Quartz' six streams produce pools of answer candidates together with numerical confidence scores. For the non-Web-based streams (*Table Lookup*, *Collection Ngrams*, *Collection Patterns* and *Tequesta*) we use Web hit counts (in a way similar to [13]) to adjust the stream's scores and thereby rerank the candidates within each pool. Apart from boosting correct answers this allows us to normalize the scores across the streams in order to make them comparable. However, this normalization does not take into account the fact that the initial scores are calculated differenly by each stream and have different and hard to predict ranges of possible values. The normalized candidate's score is based only on the stream's condifence and the Web frequency of the words of the question and of the answer.

4.2 Identifying Similar Answers

Next, we run a separate filtering module that mainly uses hand-coded heuristics to remove non-relevant candidates (e.g., it checks that answers to questions about person names indeed contain names, or for date questions the candidates bear temporal information).

In the next step, called *tiling*, the system tries to identify similar answers across the pools of answer candidates, using ideas similar to [8]. Two answers are considered similar if

- they are identical as strings, or
- the one is the substring of the other, or
- the edit distance between the strings is small compared to their length.

Note that according to this definition our notion of similarity is not an equivalence relation: the strings "*6th March 1863*" and "*May 1-3, 1863*" are both similar to "*1863*", but not to each other. At the moment, when selecting one representative in a class of similar answers our system breaks ties randomly.

4.3 Weighting Streams

After similar answer candidates have been identified, the six pools must be merged and the answer with the highest confidence selected. However, even after adjusting scores with Web hit counts, the range of possible values is different for different streams, and it is likely that more adjustment is needed for scores to be comparable. We considered several options here:

- adjust each candidate's score by a factor w_{stream} that depends on the stream generating the candidate;
- adjust with a factor $w_{stream,qtype}$ that depends on both the generating stream and the type of the question being answered;
- use confidence values as given by the streams, without adjustments.

Intuitively, the weight w_{stream} serves two purposes: in addition to normalizing scores across streams it can indicate the reliability of the streams in general. Setting w_{stream} higher would favor answers coming from the particular stream. Similarly, using the $w_{stream,qtype}$ adjustment we can also indicate the trustworthiness of the streams for different question types. The last option (no adjustment) corresponds to the (not unreasonable) belief that the scores of different streams are comparable "as is" since we have already used Web hit counts which might also have normalizing effect.

There are several ways to choose actual values for the stream weights. They can be made equal, assigned manually based on an analysis of the performance of the streams, or they can be learned automatically, from a training set of question/answer pairs. While the first two approaches are fairly straightforward and leave little room for variation, the *learning* of weights can be done in a variety of ways. Here, we describe a naive learning algorithm that proceeds by iteratively adjusting weights for each stream-question type pair. This is the method used in the experiments of Section 5.

Given a set of questions and correct answers, we learn the weights for the streams so as to maximize the number of questions answered correctly by the system as a whole. The algorithm starts with initial weights for all pairs (stream, question type):

$$w_{s,qt} := \frac{\#\text{ correct answers by stream } s \text{ to questions of type } qt}{\sum_{s'} \#\text{ correct answers by stream } s' \text{ to questions of type } qt}.$$

Then, for each question q of type qt, if the stream s^* found a correct answer a^*, but with the current stream weights the answer selection module chooses an incorrect answer, the weight $w_{s^*,qt}$ is increased so that the correct answer is selected. Let $score_s(q,a)$ denote the confidence score assigned by the stream s to an answer candidate a for a question q. Then the weight of the stream s^* is adjusted as

$$w_{s^*,qt} := w_{s^*,qt} \cdot \frac{\max_{s,a} score_s(q,a) \cdot w_{s,qt}}{score_{s^*}(q,a^*) \cdot w_{s^*,qt}}.$$

The weights of the other streams remain the same. Then all the weights are normalized so that $\sum_s w_{s,qt} = 1$. It is easy to see that after this adjustment

the answer a^* will have the highest weighted score and thus will be selected as the final answer. This procedure is repeated for all questions and then several times for the whole training set. Although this algorithm does not find globally optimal weights and even does not necessarily converge (we chose the number of iterations empirically, so that it gives the best performance on the set of training questions), our experiments showed that it does increase substantially the number of correct answers produced by the system.

Clearly, standard, more sophisticated and better understood machine learning techniques could also be applied to the task of learning optimal weights. However, one of our aims was to see whether the idea of learning stream/question type weights could be made to work in a straightforward way; it appeared to be easier to implement the naive and intuitively clear algorithm outlined above rather than squeeze the task into any of the well-known but more complicated methods. While our simple approach gives encouraging performance improvement, in future work we plan to investigate other, classical techniques.

4.4 Creating the Final Pool

Once the confidence scores of all answer candidates have been adjusted, there are still a number of ways to create a single pool of candidates:

- similar answers (as identified during tiling) are merged and their confidence scores added;
- similar answers are merged and those answers are favored that come from a larger number of streams.

As pointed in Section 2, the second approach was used in [4] for a question answering system consisting of two streams: collection- and Web-based. The underlying assumption was that similar answers coming from very different sources are more reliable. The extension of the technique to many sources may allow us to use in full the redundancy of the multi-stream architecture.

5 Experiments

Our aim was to systematically evaluate the effect of different options for answer selection in our heterogeneous QA system. Here is a short summary of the options we considered:

- use tiling or not;
- use weights based on stream (w_s) or based on stream and question type ($w_{s,qt}$);
- the choice of weights: equal, manually assigned or automatically learned from past experience;
- exploit or not the redundancy in full: consider only those answer candidates that come from the greatest number of streams, and among those pick the one with the highest final score.

For the purposes of our experiments we also created stream/question type weights manually. We analyzed the candidates of Quartz's streams for the 500 TREC 2002 questions and assigned a confidence value (0, 2, 5 or 10) to each stream and question type, based on an intuitive understanding of how good the candidates were. Observe that since these 500 questions also constituted more than a half of our training/testing collection (see below), the results for the manual voting on which we report below might be an over-estimate.

For a proper evaluation of different answer selection schemes we took the set of factoid questions from TREC 2002 and 2003 question answering tracks (913 questions) together with the patterns of correct answers provided by NIST. Since one of the options involves learning and, moreover, since our aim was to understand the *significance* of the differences, we randomly split the question set into training and evaluation sections (180 question for evaluation, the rest for training) and repeated the experiments 50 times with different splits. The splits of the question sets had to satisfy the following constraint: for all question types 80% of the questions are in the training set.

We evaluated the following variants of the answer selection module:

`ET` equal weights, tiling
`MT` manual weights, tiling
`AT` automatically learned weights, tiling
`A` automatically learned weights, no tiling
`ATU` automatically learned weights with weights not dependent on question type, tiling
`ATB` automatically learned weights, only answers from the largest number of streams are left, tiling
`ETB` equal weights, only answers from the largest number of streams are left, tiling.

We used a one-tailed t-test to establish the statistical significance of the differences observed in our experiments.

6 Results

Table 2 shows the evaluation results for the seven answer selection schemes: the average number of correct answers on a set of 180 randomly chosen questions after training on the remaining 733, measured after 50 iterations. It also shows differences between several voting schemes.

Not surprisingly, the answer selection scheme using stream/question type weights, tiling and stream redundancy (`ATB`) improves significantly over the simple baseline scheme (`ET`: with tiling but no weights) and it allows our system to give over 30% more correct answers. Moreover, both stream weighting (`AT` over `ET`) and exploiting the system's redundancy (`ETB` over `ET`) alone make significant improvements as well.

Tiling gives better performance (`AT` over `A`), although it sometimes results in answers that would have been judged *inexact* rather than *correct* by human

Table 2. Comparison of answer selection methods, measuring the average number of correct answers out of 180 randomly chosen questions (50 iterations).

Baseline		Modifications			
Scheme	#answers	Scheme	#answers	%change	Significance
ET	39.9	MT	52.0	+30.3%	** ($p < 0.001$)
		AT	50.8	+27.3%	** ($p < 0.001$)
		ATB	**52.6**	**+31.8%**	** ($p < 0.001$)
A	43.4	AT	50.8	+17.1%	** ($p < 0.001$)
ATU	48.2	AT	50.8	+5.4%	** ($p < 0.001$)
AT	50.8	ATB	52.6	+3.5%	* ($p < 0.05$)
ET	39.9	ETB	48.8	+22.3%	** ($p < 0.001$)

assessors. E.g., for question *1912. In which city is the River Seine?* our system returns the answer *Paris Opera House* rather than *Paris* only because of the tiling. In the TREC 2003 QA track over 25% of the "not wrong" answers produced by our system were judged *inexact*. While tiling does indeed help to boost strings containing correct answers, more informed filtering and type checking need to be done in the end.

The difference between AT and ATU (weighting depends on question type or not) indicates that it does indeed make sense to look at the type of the question during final answer selection, rather than simply basing the decision on the overall performance of the streams. This is explained by the fact that the streams show very different accuracy on different question types.

It is interesting to note that there is no significant difference between selection schemes with manually assigned weights (MT) and with weights chosen so as to optimize the performance on a training set of questions (AT). This is an important point as it demonstrates that there is no need to manually analyze the performance of the streams on different questions, which is a very laborious process given the number of streams and different question types. Even a very naive learning method can produce a set of weights that yields an end-to-end performance of the question answering system that equals the performance obtained with manual assignment of weights.

7 Conclusions

We have described experiments with different methods for answer selection in a multi-stream question answering system. We examined the impact of local tiling techniques, assignment of weights to streams based on past performance and/or question type, as well redundancy-based ideas. Our main finding is that redundancy-based ideas in combination with naively learned stream weights conditioned on question type work best, and improve significantly over baselines.

In future work we plan to apply other, better understood machine learning techniques to the task of learning optimal weights, and use more informed and reliable methods for answer tiling and answer filtering.

Summing up, open domain question answering systems are becoming more and more complex, increasingly relying on multiple approaches and multiple external resources. Since we do not make any assumptions about the nature of the streams, or even about the types being used in question classification, we believe that the methods and results that we have presented are widely applicable in open domain question answering.

Acknowledgments

We want to thank Gilad Mishne for help and advice, and we are grateful to our referees for useful comments. This research was supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. Maarten de Rijke was also supported by NWO under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, 612.000.207, and 612.066.302.

References

1. J. Chu-Carrol, J. Prager, C. Welty, K. Czuba, and D. Ferrucci. A multi-strategy and multi-source approach to question answering. In *Proceedings of TREC 2002*, pages 281–288, 2003.
2. C.L.A. Clarke, G.V. Cormack, G. Kemkes, M. Laszlo, T.R. Lynam, E.L. Terra, and P.L. Tilker. Statistical selection of exact answers. In *Proceedings of TREC 2002*, pages 823–831, 2003.
3. W.B. Croft. Combining approaches to information retrieval. In W.B. Croft, editor, *Advances in Information Retrieval*, pages 1–36. Kluwer Academic Publishers, 2000.
4. G. de Chalendar, T. Dalmas, F. Elkateb-Gara, O. Ferret, M. Hurault-Plantet B. Grau, G. Illouz, L. Monceaux, I. Robba, and A. Vilnat. The Question Answering System QALC at LIMSI, Experiments in Using Web and WordNet. In *Proceedings of TREC 2002*, pages 407–416, 2003.
5. M. Banko et al. AskMSR: Question answering using the Worldwide Web. In *Proc. EMNLP 2002*, 2002.
6. M. Fleischman, E.H. Hovy, and A. Echihabi. Offline strategies for online question answering: Answering questions before they are asked. In *Proceedings ACL 2003*, pages 1–7, 2003.
7. E.A. Fox and J.A. Shaw. Combination of multiple searches. In *Proceedings TREC-2*, pages 243–252, 1994.
8. M. Greenwood, I. Roberts, and R. Gaizauskas. The University of Sheffield TREC 2002 Q&A system. In *Proceedings of TREC 2002*, pages 823–831, 2003.
9. D. Hiemstra, W. Kraaij, R. Pohlmann, and T. Westerveld. Translation resources, merging strategies, and relevance feedback for cross-language information retrieval. In *Proceedings CLEF 2000*, pages 102–115. Springer, 2001.
10. E. Hovy, H. Hermjakob, M. Junk, and C.-Y. Lin. Question answering in Webclopedia. In *Proceedings TREC-9*, 2000.
11. V. Jijkoun, J. Kamps, G. Mishne, C. Monz, M. de Rijke, S. Schlobach, and O. Tsur. The University of Amsterdam at TREC 2003. In *TREC 2003 Notebook Papers*, 2003.
12. V. Jijkoun, G. Mishne, and M. de Rijke. How frogs built the Berlin Wall. In *Proceedings CLEF 2003*. Springer, to appear.

13. B. Magnini, M. Negri, R. Prevete, and H. Tanev. Is it the right answer? Exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 425–432, 2002.
14. D. Moldovan, S. Harabagiu, M. Paşca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus. The structure and performance of an open domain question answering system. In *Proceedings ACL 2000*, pages 563–570, 2000.
15. D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21:133–154, 2003.
16. C. Monz and M. de Rijke. Tequesta: The University of Amsterdam's textual question answering system. In *Proceedings TREC 2001*, pages 519–528, 2002.
17. J. Prager, E. Brown, A. Coden, and D. Radev. Question-answering by predicitive annotation. In *Proceedings SIGIR 2000*, pages 184–191, 2000.
18. M.M. Soubbotin and S.M. Soubbotin. Use of patterns for detection of likely answer strings: A systematic approach. In *Proceedings TREC 2002*, pages 325–331, 2003.
19. Text REtreival Conference (TREC). URL: `http://trec.nist.gov`.
20. J. Xu, A. Licuanan, J. May, S. Miller, and R. Weischedel. TREC 2002 QA at BBN: Answer Selection and Confidence Estimation. In *Proceedings of TREC 2002*, pages 96–101, 2003.