The University of Amsterdam at the TREC 2003 Question Answering Track

Valentin Jijkoun Gilad Mishne Christof Monz*
Maarten de Rijke Stefan Schlobach Oren Tsur†
Language & Inference Technology Group
University of Amsterdam
http://lit.science.uva.nl/

Abstract: We describe our participation in the TREC 2003 Question Answering track. We explain the ideas underlying our approaches to the task, report on our results, provide an error analysis, and give a summary of our findings so far.

1 Introduction

The aim for our participation in the Question Answering track at TREC 2003 was to experiment with a new multi-stream architecture, in which we implemented 6 separate subsystems that each try to answer questions in different ways. We also wanted to experiment with a dedicated biography question module that is currently in development.

Our experiments exploited the home-grown FlexIR document retrieval system [9]. The main goal underlying FlexIR's design is to facilitate flexible experimentation with a wide variety of retrieval components and techniques; we used FlexIR's implementations of the `Lnu.ltc` weighting scheme, various language models, as well as the Okapi scheme.

Current Question Answering (QA) systems, as reflected by the TREC QA track participants, can be divided into two categories: *knowledge-intensive* systems, that make use of various linguistic tools for the question answering process, and *redundancy-based* systems, that rely on very high volumes of data (in many cases, the Web) and take a more shallow approach to text analysis. Until last year, we were focused on the first approach, concentrating our QA efforts exclusively on *Tequesta* [10, 11]. This approach may be successful for some types of questions, but for others more shallow approaches seem more beneficial. This year we expanded our QA work and implemented a *multi-stream* approach. While maintaining Tequesta as one of the approaches, we developed additional systems that compete which each other to find the correct answer. These systems, or "streams," employ a range of redundancy-based and knowledge-intensive techniques. We took part in the main QA task and in the passage QA task. For our participation in the main task we employed our new multi-stream architecture; for the passage task we relied on the Tequesta stream only.

The rest of this paper is organized as follows. In two (largely self-contained) sections we describe our work for the main task and the passage task. Finally, we summarize our findings in a concluding section.

2 The Main Task

2.1 System Description

We now describe the approach we adopted for the main QA task; we devote separate subsections to factoid questions on the one hand, and list questions and definition questions on the other. The system consists of 6 separate QA streams and a final answer selection module that combines the results of all streams and produces the final answers. An important benefit of this architecture is easy modification, maintenance, and testing of the dif-

*Now at the Institute for Advanced Computer Studies, University of Maryland, 3161 A.V. Williams Building, College Park, MD 20742, USA. Email: christof@umiacs.umd.edu.

†Now at Bar-Ilan University, Ramat Gan, Israel. Email: tsuror@cs.biu.ac.il.

ferent subsystems as well as easy integration of multiple sources of information. Evaluation of the contribution of each stream to the entire QA process becomes a relatively simple task too. We now describe the streams.

Table Lookup. This stream uses specialized knowledge bases constructed by preprocessing the collection, similar in spirit to [4]. 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. When a question type indicates that the question might potentially have an answer in these tables, a lookup is performed in the appropriate table and answers found are assigned high confidence.

We hand-crafted a small number of regular expressions for extracting information about the categories listed in Table 1. For instance, the "Location" category concerns geographic information of the following type "Amu Darya, river, Turkmenistan, XIE19990811.0277," where the first field indicates a location, the second its type, the third a country or region in which it is located, and the fourth the identifier for the document from which it was extracted. "Geography" contains similar information, but without the type; "Leaders" has information of the following kind "Dutch, Foreign Minister, Jozias van Aartsen, XIE19991027.0270", and "Roles" generalizes this to also include other roles besides government-related ones.

Table 1: Facts extracted from the AQUAINT corpus.

Category	# Facts	Category	# Facts
Abbreviations	31737	Birthdates	9156
Capitals	1273	Currencies	231
Dates	9331	Deathdates	1510
Geography	70363	Height	15603
Inhabitants	2025	Languages	853
Leaders	18073	Locations	1348
Manners of death	857	Organizations	98758
Roles	396558		

When a question is classified as possibly having an answer in a table, we first identify the question keywords that will be used in the table search. Next, a line matching all of the words in the order they appeared in the question is searched; if no line matches, we look again for a line containing all words, this time in any word order. If there is still no match, we start removing words from the list of words to match; the order of removal is based on the frequency of words in the language (i.e., common words are removed first) and part-of-speech tags (e.g., superlatives like *fastest*, *largest* are removed last). We do this until some threshold is reached (percentage of lookup words out of total keywords in the question). When a matching line is found, we return the text in the column that is declared to contain the information required as the answer.

Pattern Matching. 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?* will possibly match the pattern `<Capitalized-Words>(,| is) the richest country in the world`. In these cases, the position of the answer within the context is also known when generating the context pattern; in the given example, it would be the capitalized word or words (and indeed, in document XIE19980302.0146, this pattern matches against "... Although the United States is the richest country in the world, 20 percent of its population ...").

The Pattern Matching stream consists of three stages: *Generation*, *Document Prefetch* and *Matching*. In the Generation stage, the question is analyzed and possible answer patterns are generated. For questions like *2347. Where is Mount Olympus?* the question type and focus (both provided by the question classifier) are sufficient for generating a number of answer patterns. For other questions (e.g., *2375. What date did Thomas Jefferson die?*) we also use a set of manually created rules based on part-of-speech tags of the question words and a dictionary of word forms, in order to rewrite the question into declarative forms (e.g., `Thomas Jefferson (died|dies) (on|in) <answer>`). In the Prefetch stage, for each generated pattern a query containing words from it is formed, and documents are retrieved from the collection using the query. In the final stage, the patterns are matched against the retrieved documents, and answers are extracted from the matches.

Two variations of this stream were implemented, *Web Pattern Matching* and *Collection Pattern Matching*. For the first variation the text collection was the Web, and for the second, the local AQUAINT corpus. For the prefetch stage we used the top-ranking documents from Google (for the Web variation) and all matching documents retrieved using a boolean query to our document retrieval engine FlexIR against the AQUAINT corpus.

Ngram Mining. This stream, similar in spirit to [2, 3], constructs a weighted list of queries for each question using a shallow reformulation process, similar to the Pattern Match stream. The queries are then sent to a large document collection; we implemented two variations for this stream, *Web Ngram Mining* and *Collection Ngram Mining*, using the Web and the local AQUAINT corpus, respectively. For Web searches, we used Google, and for local searches FlexIR, with the `Lnu.ltc` weighting scheme. Then, we looked at word ngrams in the relevant retrieved document paragraphs (for the Web we used the snippets provided by Google, and for the collection we used a window of 200 bytes around the query). The ngrams were ranked according to the weight of the query that generated them, their frequency in the paragraphs, their NE type, the proximity to the query keywords and more parameters, and the top-ranking ngrams were considered answer candidates. To find justification for the answer in the local corpus, we constructed a query with keywords from the question and the answer, and considered the top-ranking document for this query to be the justification, this time using an Okapi model as this tends to do well on early high precision in our experience.

Tequesta. As mentioned before, this is a stream that implements a linguistically informed approach to QA. We defer a discussion of this stream to Section 3 where we describe our strategy for the passage task.

Many components are shared by all streams, including a locally developed named entity tagger and the following:

Question Classifier. An incoming question is first analyzed for its type (e.g., `date-of-birth`), expected answer type (e.g., `location`) and focus (the *core* of the question, used e.g., for answer pattern generation). Currently our system recognizes 37 question types. The question analysis is based on surface and part-of-speech patterns. We also use hierarchical relations in WordNet to identify semantic classes of question focus words (e.g., this allows us to assign the type `person-ident` to the question *1943. What is the name of Ling Ling's mate?*).

Web Ranking. The answer candidates produced by the streams have different confidence levels, generated by stream-specific parameters and measuring methods. To compare these levels, a uniform way of ranking the candidates was required. To this end we implemented a search engine hit count module, similar to [6].

Answer Selection. Each of our streams produces a pool of answer candidates, with normalized confidence scores. After filtering the candidates to remove obvious noise, we create a joint pool of answers, adjusting each candidate's score by a factor that reflects the past performance of its stream on questions of the same type. We tried different ways of assigning these stream/question-type weights: manually (i.e., based on human intuition about how good different streams perform on different questions) and automatic (using Machine Learning to find weights that optimize the performance of the system on a training set of questions) [5]. In the joint pool of answer candidates we identify identical or similar (small edit distance) answers, merge and add their confidence scores. Finally, a candidate with the highest score is returned.

List and Definition Questions

Because of time constraints, we were unable to implement a proper module for handling list questions. All list questions were automatically rewritten into factoids using rule-based transformations (e.g., *2097. Which countries were visited by first lady Hillary Clinton?* was transformed to *Which country was visited by first lady Hillary Clinton?*) and fed to our multi-stream QA system. The top *N* candidate answers to this factoid question were submitted as answers to the original list question. We experimented with different values of *N* (10 and 20 in our official runs) and with different numbers of retrieved documents used during answer selection (both for collection- and web-based QA streams).

In contrast to list questions, we did invest a serious effort in developing a component for handling definition questions. More precisely, we piggybacked on ongoing inhouse activities aimed at developing a QA system for handling "biography oriented" definitions on the web [13]. The main steps in our handling of definition questions are Question Analysis (very similar to the analysis carried out for factoids), Answer Retrieval (always from external resources), Answer Filtering, and Answer Justification (very similar to the justification performed for externally found answers to factoid questions).

For concept definition questions we followed a

WordNet-based strategy as discussed in the literature [12]. Given a question that asks for a definition of a concept, we simply consult WordNet. As our primary strategy for handling person definition questions, we also consulted an external resource. The main resource used is `biography.com`. However, in many cases no biography could be found in this resource. In such cases we backed off to using Google, with queries obtained by combining the name of the person in question with varying subsets of a predefined set of hand-crafted features (including "born", "graduated", "suffered", etc.) For questions asking for definitions of organizations the latter was the strategy used (with a set of "organization features").

As a final fallback option for each type of definition question, if the use of the strategies mentioned earlier returned no satisfactory results, we simply submitted `<question term> is a` to Google and mined the snippets returned. This method worked surprisingly well for questions like *2385. What is the Kama Sutra?*.

Given a set of candidate answer snippets, we performed two more steps before carrying out the final answer justification step: we separated junk snippets from valuable snippets and we identified snippets whose content is very similar. We addressed the first step by analyzing the distances between query terms submitted to the search engine and the sets of features, and by means of shallow syntactic aspects of the different features such as sentence boundaries. To address the second step we developed a snippet similarity metric based on edit distance, stemming, stopword removal, and keyword overlap.

2.2 Runs

We submitted 3 runs. These runs used the exact same strategies and settings for definition questions. They did differ in their settings for factoids and list questions. Here's a brief description:

`UAmsT03M1` For factoids, the answer selection module used automatically learned stream/question weights; answers coming only from external sources (streams based on Web) were justified against the AQUAINT collection using the Okapi model. For each list question the top 10 answers to its factoid counterpart were submitted.

`UAmsT03M2` For factoids, the weights for answer selection were learned automatically; external answers were discarded. For list questions the number of collection and web documents used for answer mining was increased, and the top 20 answers were submitted for each question.

`UAmsT03M3` Manually assigned weights were used for answer selection; external answers were discarded. The number of documents for answering list questions was as in `UAmsT03M2`, but only top 10 answers were submitted.

Our three runs allowed us to compare the impact of justification, and the impact of using manually assigned versus learned weights for our answer selection. For the list questions we wanted to evaluate the effect of *using more data* and of *giving more answers* on the final performance.

2.3 Results

Table 2 gives the detailed results of our system for the 413 factoid questions: accuracy and the number of correct (R), unsupported (U), inexact (X) and wrong (W) answers.

Table 2: Results for the QA track (factoid questions).

Run identifier	Accuracy	R	U	X	W
`UAmsT03M1`	0.136	56	22	32	303
`UAmsT03M2`	0.145	60	20	26	307
`UAmsT03M3`	0.128	53	24	30	306

2.4 Error Analysis

Analyzing errors made by a QA system is a complex task. In [7], such an analysis is based on examination of the outputs of every module in the process separately, and attributing the error to the first malfunctioning module. In many cases an error in one of the earlier stages of the QA pipeline (for example, the question classification module) does indeed cause cascaded errors later. But when diagnosing a system with multiple independent approaches such as ours, this does not necessarily hold; we found incorrectly classified questions which were still answered correctly due to the redundancy-based modules, and many other counter-examples to the "cascaded errors" assumption. Therefore, we chose to examine the incorrect answers produced by the system, and associate each of them

with a *main error type*, the dominant reason for producing this incorrect answer.

Table 3 shows the most common error types for run `UAmsT03M1`; for each incorrectly answered question (counting inexact or unsupported answers as incorrect), we examined the candidate answer list produced by our system (the list contains less than 10 candidates on average). If the correct answer was in this list, we classified the error types of the candidates that received a higher rank than it; otherwise, we classified the top 3 ranking candidates. Since we examined more than one answer per question, there may be multiple error types for a specific question.

Table 3: Frequent Error Types in **UAmsT03M1**.

Error Type	**Frequency**
Answer Selection	134 (38%)
Named-Entity	78 (22%)
Question Classification	67 (19%)
Justification	58 (16%)
Unit Error	53 (15%)

A brief explanation of main error types follows:

- *Answer Selection* errors describe an incorrect answer with the correct named entity or concept type which typically appears in relevant documents. An example is the answer `George Bush` for *2391. What president created social security?* – the answer type is correct, and is very frequent in relevant documents (both in the local collection and on the web).
- *Named-Entity* errors result from an incorrect classification of a phrase as a named entity which matches the expected answer type. An example is the answer `Springsteen` for *2001. What rock band sang "A Whole Lotta Love"?*.
- *Question Classification* errors are cascaded errors originating from an incorrect question type assigned to the question at an early stage of the QA pipeline, or failure to assign any question type to it.
- *Justification* errors are correct answers which were obtained using external resources, and were not projected correctly to the local corpus.
- *Unit Errors* are answers of the correct named entity type, but incorrect granularity (i.e. state instead of city) or out of range (according to world-knowledge). An example is the answer `about 2 billion dollar` for *2302. How much did the first Barbie cost?* (referring to profits rather than costs).

While our analysis revealed many technical issues that need to be addressed – such as over-tiling of ngrams, resulting in inexact answers (`Colombia country South America` instead of `Colombia`) – most of the errors stem from the shallow answer-selection techniques used by our system. We currently use mostly frequency counts and proximity measures to select the answer candidates; this works for questions which have a large amount of relevant documents, but for other questions deeper analysis is required. An alternative approach, still relying on redundancy methods, is to expand the number of retrieved documents using query expansion methods (both for the local corpus and the web) – an approach which is also almost not used in our system. Our main conclusion is that while we continue to see redundancy-based methods as our basic strategy for QA, shallow NLP and reasoning methods should be selectively used throughout the process, especially when the number of retrieved documents is low.

A few more remarks are worth making. First, although the run with the automatically learned weights for answer selection from multiple streams (`UAmsT03M2`) outperformed the run with manually assigned weights (`UAmsT03M3`), our subsequent experiments revealed that whereas a small difference exists, it is not statistically significant. However, both runs improve significantly over a baseline system with equal weights to all streams.

We also evaluated the contribution of different streams to the performance of the system on the factoids (using unofficial answer patterns). Table 4 gives the results (the number of "correct" answers, i.e., those that match the patterns) for the whole system, for separate streams and for the system with one of the streams turned off. As expected, each of the six streams answered some questions correctly and more interestingly, each stream contributed to the overall performance of the system. The two "worst" performing streams (predictably, collection-based pattern matching and ngram mining) brought one more answer each either at the top rank or in the top 5. Surprisingly, the "winner" among the streams is equivocal: while *Table Lookup* allows the system to answer 15 questions more, *Web Ngrams* accounts for more (35 vs. 19) unique correct

Table 4: Contribution of different streams.		
Configuration	# correct	# correct in top 5
All streams	98	165
Collection ngrams	39	42
Without collection ngrams	98	164
Web ngrams	65	115
Without Web ngrams	89	130
Collection patterns	39	39
Without collection patterns	97	165
Web patterns	51	59
Without Web patterns	94	163
Table lookup	71	77
Without table lookup	83	146
Tequesta	63	102
Without Tequesta	91	140

answer candidates in the top 5.

Table 5 gives the combined results for the 3 QA tasks (accuracy for factoids, F score for list and definition questions) and the final scores of our runs. The results for the

Table 5: Results for the QA track.				
Run identifier	A (Fact)	F (List)	F (Def)	Overall
UAmsT03M1	0.136	0.054	0.315	0.160
UAmsT03M2	0.145	0.042	0.308	0.160
UAmsT03M3	0.128	0.035	0.292	0.146

list questions suggest that using more retrieved documents for answer extraction and submitting more answer candidates hurts performance: the increase in recall does not compensate for the drop in precision.

Turning to definition questions now, recall that there is *no* difference between the three runs listed in Table 5 as far as definition questions are concerned, despite the different scores in the table. The differences are due to inconsistencies in the judgments provided by NIST. Table 6 provides a breakdown of the scores for the different types of definition questions; the highest scores are obtained for person definitions, which reflects the fact that those are the type of definition questions in which we put most work. As an aside, in our submission we found *no* answer

Table 6: Breakdown of F scores for definition questions.				
Run identifier	Concept	Person	Org.	Overall
UAmsT03M1	0.150	0.392	0.268	0.315

for 19 of the 50 definition questions. If we compute the F score not over all 50 question but only over questions with a positive F score, we obtain an average of 0.527. In post-submission experiments we changed the subsets of features we use in the queries sent to Google as well as the number of queries/subsets we use. The snippets-similarity threshold was also tuned in order to filter more snippets. This resulted in a reduction of unanswered definition questions to 6 instead of 19. Using our own (unofficial) assessment, this yielded an F score of 0.688. Those changes also reflected in the average answer length. After the parameters tuning the average length was half of the average TREC submission answer, improving precision and contributing to the F score.

2.5 Conclusions for the Main Task

Our general conclusion on answering factoid questions is that our new multi-stream approach helped answer considerably more questions than our "old" single-stream Tequesta system. This year's questions seem *much* harder than those of previous years. A preliminary error analysis shows that retrieval, named entity recognition, and answer selection all require further attention. Our main conclusion on answering definition questions is that external dictionary-like resources are crucial, but a feature-based approach offers an effective strategy if such resources are absent or too sparse. Following an analysis of our TREC results, we investigated the use of trainable text classifiers as a pre-processing stage instead of the features vectors, treating the web as a 'noisy' external knowledge source, and using the text classifier to filter out the noise. Initial results show that using text classifiers greatly improves the F score and the coherence of answers.

3 The Passage Task

The aim of the passage task was to return an excerpt from a document rather than an exact answer. Excerpts had to be unmodified snippets from a document in the AQUAINT collection, and were not allowed to be longer than 250 characters. For the passage task only the factoid questions from the main task were used, i.e., list and definition questions were not included.

3.1 System Description

For the passage task, we used a modification of the Tequesta question answering system, which has remained largely unchanged since TREC 2002 [10, 11]. We dropped the exact answer output feature, and included some of the context surrounding the answer identified by Tequesta. We added the use of minimal span weighting for identifying documents that are likely to contain an answer to a given question. We used minimal matching spans as the snippets in which to find the exact answer.

Minimal span weighting takes the positions of matching terms into account, but does so in a more flexible way than passage-based retrieval; see [8] for details. Intuitively, a minimal matching span is the smallest text excerpt from a document that contains all terms which occur in the query and the document. More formally, given a query q and a document d, the function $\text{term\_at\_pos}_d(p)$ returns the term occurring at position p in d. A *matching span* (ms) is a set of positions that contains at least one position of each matching term, i.e. $\bigcup_{p\in \text{ms}} \text{term\_at\_pos}_d(p) = q \cap d$.

Then, given a matching span ms, let b_d (the beginning of the excerpt) be the minimal value in ms, i.e., $b_d = \min(\text{ms})$, and e_d (the end of the excerpt) be the maximal value in ms, i.e., $e_d = \max(\text{ms})$. A matching span ms is a *minimal matching span* (mms) if there is no other matching span ms$'$ with $b'_d = \min(\text{ms}')$, $e'_d = \max(\text{ms}')$, such that $b_d \neq b'_d$ or $e_d \neq e'_d$, and $b_d \leq b'_d \leq e'_d \leq e_d$.

Minimal span weighting depends on three factors.

1. *document similarity*: The document similarity is computed using the Lnu.ltc weighting scheme Buckley et al. [1] for the whole document. Similarity scores are normalized with respect to the maximal similarity score for a query.
2. *span size ratio*: The span size ratio is the number of unique matching terms in the span over the total number of tokens in the span.
3. *matching term ratio*: The matching term ratio is the number of unique matching terms over the number of unique terms in the query, after stop word removal.

The msw score is the sum of two weighted components: the normalized original retrieval status value (RSV), which measures *global similarity* and the spanning factor which measures *local similarity*. Given a query q, the original retrieval status values are normalized with respect to the highest retrieval status value for that query:

$$\text{RSV}_n(q,d) = \frac{\text{RSV}(q,d)}{\max_d \text{RSV}(q,d)}.$$

The spanning factor is the product of two components: the span size ratio, which is weighted by α, and the matching term ratio, which is weighted by β. Global and local similarity are weighted by λ. The optimal values of the three parameters λ, α, and β were found to be $\lambda = 0.4$, $\alpha = 1/8$, and $\beta = 1$ by empirical means. Parameter estimation was done using the TREC-9 data collection only, but it proved to be the best parameter setting for all collections.

The final retrieval status value (RSV') based on minimal span weighting is defined as follows, where $|\cdot|$ is the number of elements in a set: If $|q \cap d| > 1$ (that is, if the document and the query have more than one term in common), then

$$\text{RSV'}(q,d) = \lambda \cdot \text{RSV}_n(q,d) + (1-\lambda) \cdot \left(\frac{|q \cap d|}{1 + \max(mms) - \min(mms)} \right)^{\alpha} \cdot \left(\frac{|q \cap d|}{|q|} \right)^{\beta}.$$

If $|q \cap d| = 1$ then RSV'(q,d) = $\text{RSV}_n(q,d)$.

Given a minimal matching span, the document analysis component of Tequesta tries to identify a phrase which is of the appropriate type. All phrases that are of the appropriate type are considered candidate answers. Tequesta selects answers by considering the frequency of a candidate answer and relying on linking a candidate answer to the question by proximity. Hence, all candidate answers are weighted equally. But there is one exception. If the question is of type `what-np`, candidate answers that are in a WordNet hypernym relationship with the question focus receive a higher weight than candidate answers that are identified by means of the fallback strategy.

Once an answer has been selected, the corresponding minimal matching span from which the answer has been extracted is returned as the answer passage, trimmed down to 250 characters if necessary.

3.2 Runs, Results and Conclusion for the Passage Task

We submitted one run to the passage task, run id `UAmsT03P1`. The results are shown in Table 7. (R) stands

for passages that contained a correct and exact answer, (U) for passages that contained the correct answer, but were not supported by the corresponding document, and (W) stands for wrong answers. The passage track does

Table 7: Results for the QA passage track

Run identifier	Accuracy	R	U	W
UAmsT03P1	0.111	46	6	361

not make a distinction between exact and inexact (X) answers, as in the main task. Here, an inexact answer is simply judged wrong (W).

The results were quite disappointing. At this point we are not sure what caused this rather bad performance. Before submitting this year's run to the passage track, we conducted some experiments on the question sets from previous TRECs, and these results were substantially better. Therefore, one explanation could be that this year's question set was much harder than the previous ones, but a more detailed error analysis remains to be done.

4 Conclusions

We have described our participation in the TREC 2003 Question Answering Track. This year, our work was largely motivated by our move to a new, multi-stream architecture. Although a further and more detailed analysis of the performance of the system remains to be done, our preliminary results show that different approaches to the QA process do produce answers to different question types. Our combined use of external resources and handcrafted feature sets proved to be a successful approach for answering definition questions.

Acknowledgments

Thank you to Börkur Sigurbjörnsson for useful suggestions and discussion. We thank Raffaella Bernardi for help on the question classifier and table extraction components, and Karin Müller and Detlef Prescher for help with our named entity recognizer. Many thanks to Henry Chinaski for sanity checks and inspiration.

Valentin Jijkoun, Gilad Mishne, and Stefan Schlobach were supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. Christof Monz was supported by NWO under project numbers 612-13-001 and 220-80-001. Maarten de Rijke was supported by NWO under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, 220-80-001, 612.000.207, and 612.066.302.

References

[1] C. Buckley, A. Singhal, and M. Mitra. New retrieval approaches using SMART: TREC 4. In *The Fourth Text REtrieval Conference (TREC-4)*, 1996.

[2] C. Clarke, G. Cormack, and T. Lynam. Exploiting redundancy in question answering. In D. H. Kraft, W. B. Croft, D. J. Harper, and J. Zobel, editors, *Proceedings of SIGIR 2001*, pages 358–365, 2001.

[3] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In P. Bennett, S. Dumais, and E. Horvitz, editors, *Proceedings of SIGIR 2002*, pages 291–298, 2002.

[4] M. Fleischman, E. H. Hovy, and A. Echihabi. Offline strategies for online question answering: Answering questions before they are asked. In *Proceedings of ACL 2003*.

[5] V. Jijkoun and M. de Rijke. Answer selection in a multi-stream open domain question answering system. In *Proceedings of ECIR'04*, 2004.

[6] B. Magnini, M. Negri, R. Prevete, and H. Tanev. Is it the right answer? exploiting web redundancy for answer validation. In *Proceedings of ACL 2002*, pages 425–432, 2002.

[7] D. Moldovan, M. Pasca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Trans. Inf. Syst.*, 21(2): 133–154, 2003.

[8] C. Monz. *From Document Retrieval to Question Answering*. PhD thesis, University of Amsterdam, 2003.

[9] C. Monz and M. de Rijke. Shallow morphological analysis in monolingual information retrieval for Dutch, German and Italian. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Proceedings CLEF 2001*, 2002.

[10] C. Monz and M. de Rijke. Tequesta: The University of Amsterdam's textual question answering system. In Voorhees and Harman [14], pages 519–528.

[11] C. Monz, J. Kamps, and M. de Rijke. The University of Amsterdam at TREC 2002. In E. M. Voorhees and L. P. Buckland, editors, *The Eleventh Text REtrieval Conference (TREC 2002)*, pages 603–614, 2003.

[12] J. Prager, J. Chu-Carroll, and K. Czuba. Use of Wordnet hypernyms for answering what-is questions. In Voorhees and Harman [14], pages 250–257.

[13] O. Tsur. Definitional question answering using trainable classifiers. M.Sc thesis, University of Amsterdam, 2003.

[14] E. M. Voorhees and D. K. Harman, editors. *The Tenth Text REtrieval Conference (TREC 2001)*, 2002.