Making Stone Soup

Evaluating a Recall-Oriented Multi-Stream Question Answering System for Dutch

David Ahn   Valentin Jijkoun   Karin Müller
Maarten de Rijke   Stefan Schlobach*   Gilad Mishne

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
E-mail: {ahn,jijkoun,kmueller,mdr,schlobac,gilad}@science.uva.nl

Abstract. We describe the participation of the University of Amsterdam in the Question Answering track at CLEF 2004. We took part in the monolingual Dutch task and, for the first time, also in the bilingual English to Dutch task. This year's system is a further elaboration and refinement of the multi-stream architecture we introduced last year, extended with improved candidate answer re-ranking and filtering, and with additional answer finding strategies. We report the evaluation results for the whole system and its various components. The results indicate the recall-oriented approach to QA is an effective one.

1 Introduction

To address the question answering (QA) task, one has to address a challenging *recall* problem. As with many language processing tasks, we face a vocabulary gap—the phenomenon that the question and its answer(s) may be phrased in different words. For QA, the vocabulary gap can be especially challenging as systems have to return highly relevant and focused text snippets as output, given very short questions as input. To address the vocabulary gap problem, we advocate a *multi-stream* architecture which offers multiple ways of identifying candidate answers. Each stream serves as an essential ingredient to the whole system, and in this way it is reminiscent of *stone soup* (http://en.wikipedia.org/wiki/Stone_soup). This kind of approach needs an elaborate filtering and ranking mechanism to weed out incorrect candidate answers. In 2003, we completed a first version of this architecture, of which we made good use for the QA tracks both at CLEF [10] and at TREC [11]. For the 2004 edition of the QA@CLEF task, we fine-tuned and extended the architecture.

At CLEF 2004, we took part in the monolingual Dutch QA task and the bilingual English-to-Dutch QA task. For the monolingual task, the questions—factoid and definition questions—were given in Dutch and for the bilingual task, the questions were given in English. For both tasks, the answers had to be identified in the Dutch CLEF collection. Our main aim with our monolingual work was to extend and improve our QA system following an error analysis after the 2003 edition of the task. The bilingual English-to-Dutch task was new for us. We translated the questions into Dutch and

* Currently at the Division of Mathematics and Computer Science, Free University Amsterdam.

we then proceeded as in the monolingual task. Our main aim here was to evaluate the applicability of our system in a cross-language setting and to see whether correct results obtained by the bilingual run are a subset of the monolingual one—or whether something can be gained by combining them.

The paper is organized as follows. In Section 2, we describe the architecture of our QA system. Section 3 describes our official runs. In Section 4, we discuss the results obtained and give an analysis of the performance of different components of the system. We summarize and conclude in Section 5.

2 System Description

Many QA systems share the following pipeline architecture. A question is first associated with a *question type* such as DATE-OF-BIRTH or CURRENCY, chosen from a predefined set. A query is then formulated on the basis of the question, and an information retrieval engine is used to identify a list of documents that are likely to contain the answer. Those documents are sent to an *answer extraction* module, which identifies candidate answers, ranks them, and selects the final answer. On top of this basic architecture, numerous add-ons have been devised, ranging from logic-based methods [12] to ones that rely heavily on the redundancy of information available on the World Wide Web [5].

In essence, our system implements multiple copies of the standard architecture, each of which is a complete standalone QA system. The general overview of the system is given in Figure 1. Each copy shares (at least) two modules: the question classification and the answer identification module. The question classifier is based on manually developed patterns that take different types of information into account: the question word, certain classes of verbs, etc. This year, we improved our question classifier by incorporating Dutch WordNet to deal with questions such as *Which X . . . ?*, where the semantic type of *X* is now used for classification.

Each of the streams produces a ranked list of candidate answers, but not necessarily for all types of questions. The overall system's answer is then selected from the combined pool of candidates through a combination of merging and filtering techniques. We add to the answer selection procedure a type checking module which checks whether the answer is of the correct type given the expected answer type identified during question analysis. For a reasonably detailed discussion of our QA system architecture, we refer to [10, 11].

This year's system contains 8 streams organized in four groups, depending on the main data source from which they try to answer questions. The streams either consult the Dutch CLEF corpus, the English CLEF corpus, or the Web. We added one new stream to our system which consults information sources like Wikipedia. We now provide a brief description of these four groups.

2.1 Streams that Consult the Dutch CLEF Corpus

Four streams generate candidate answers from the Dutch CLEF corpus in parallel: *Lookup*, *Pattern Match*, *Ngrams*, and *Tequesta*.

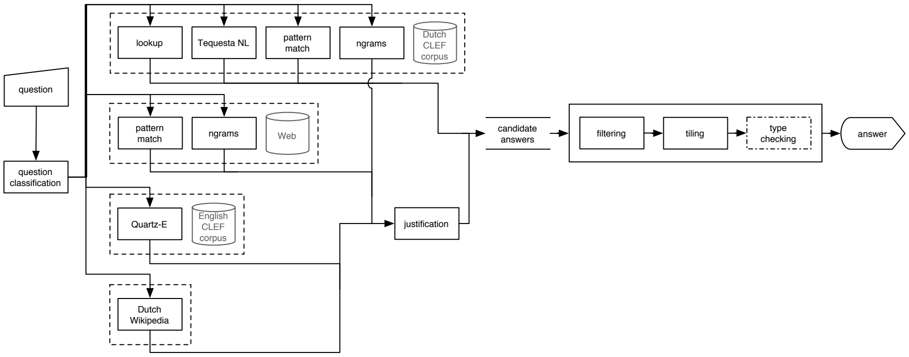

Fig. 1. Quartz-N: the University of Amsterdam's Dutch Question Answering System.

The *Table Lookup* stream uses specialized knowledge bases constructed by preprocessing the collection, exploiting 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 more or less 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 knowledge base and answers which are found there are assigned high confidence. For a detailed overview of this stream, see [9]. In addition to the knowledge bases used in CLEF 2003, we built new ones (such as AWARDS and MEASUREMENTS, storing facts about winners of various prizes and information about dimensions of objects, respectively). Furthermore, we enriched our previous knowledge bases, which were extracted using surface patterns, with information extracted with syntactic patterns from the Dutch CLEF collection parsed by the Alpino parser, a wide coverage dependency parser for Dutch [2]. Earlier experiments on the AQUAINT corpus had suggested that offline extraction using syntactic extraction patterns can substantially improve recall [8].

The *Dutch Tequesta* stream is a linguistically informed QA system for Dutch that implements the traditional architecture outlined above. Among others, it uses a Part-of-Speech tagger (a TnT-based tagger [3] trained on the *Corpus Gesproken Nederlands* [15]), our own named entity tagger for Dutch [6], as well as proximity-based candidate answer selection [13].

In the *Pattern Match* stream, zero or more regular expressions are generated for a question according to its type and structure. These patterns match strings which have a high probability of containing the answer with high probability and are used to extract such strings from the entire document collection.

The *Ngram* stream, similar in spirit to [4], constructs a weighted list of queries for each question using a shallow reformulation process, similar to the *Pattern Match* stream. These queries are fed to a retrieval engine (we used our own FlexIR[14], with the `Lnu.ltc` weighting scheme), and the top retrieved documents are used for harvesting word ngrams. The ngrams are ranked according to the weight of the query that generated them, their frequency, NE type, proximity to the query keywords and other parameters; the top-ranking ngrams are taken as candidate answers. The output of this stream is piped to the *Justification* module (see below).

As mentioned earlier, we aim at higher recall at the earlier stages, relying on various filtering mechanisms to "clean" the results later, and achieve high precision as well. Therefore, for both the *Ngram* and the *Pattern Match* streams, we extended the generated regular expressions and queries, compared to our system at CLEF 2003—sometimes creating ungrammatical ones under the assumption that possibly incorrectly extracted candidate answers would be filtered out later.

2.2 Streams that Consult the English CLEF Corpus

One of the streams used by Quartz-N is the English language version of our QA system, which consults the English CLEF corpus instead of the Dutch version (but which is otherwise similar to the Dutch version). The answers found by Quartz-E are also piped to the *Justification* module.

2.3 Streams that Consult the Web

Quartz-N also has two streams that attempt to locate answers on the web: *Ngram* and *Pattern Match*. We retrieve documents using Google: ngrams are harvested from the Google snippets, while pattern matching is done against the *full* documents retrieved. In all other respects, those two streams work the same way as the corresponding streams that consult the Dutch CLEF corpus.

2.4 Streams that Use other Resources

A new stream this year was the Wikipedia stream. Like the streams that consult the Web or the English document collection, this stream also uses an external corpus—the Dutch Wikipedia (`http://nl.wikipedia.org`), the Dutch version of an open-content encyclopedia. Since this corpus is much "cleaner" than newspaper text, the stream operates in a different manner. First, the *focus* of the question is identified—this is usually the main named entity in the question—and looked up in the encyclopedia. Then, the focus's encyclopedia entry is looked up; since Wikipedia is standardized to a large extent, the entry has a template-like form. Thus, using knowledge about the templates employed in Wikipedia, information such as DATE-OF-DEATH and FIRST-NAME can easily be extracted.

2.5 Answer Selection Procedures

While each of the above streams is a "small" QA system in itself, many components are shared between the streams, including an Answer Justification module, a Type Checking module, and a Filtering and Tiling module, all of which we will now describe.

Answer justification. As some of our streams obtain candidate answers *outside* the Dutch CLEF corpus, and as answers need to be supported, or *justified*, by a document in the Dutch CLEF corpus, we need to find justification for candidate answers found externally. To this end, we construct a query with keywords from a given question and candidate answer, and take the top-ranking document for this query to be the justification. We use an Okapi-based retrieval model as this tends to do well on early high precision in our experience. Additionally, we use some retrieval heuristics, such as marking the answer words as boolean terms in the query (requiring them to appear in retrieved documents).

Type Checking. To compensate for named entity errors made during answer extraction, our type checking module (see [16] for details) uses WordNet and several geographical knowledge bases to remove candidates of incorrect type for location questions. Since the resources used by the type checker are English, some adaptation for the Dutch language was needed. The question target of a Dutch question is extracted and automatically translated into English. Candidate answers are also translated, and then the method described in [16] is applied to check whether the candidates match the expected answer type.

Filtering and Tiling. A detailed error analysis carried out after the 2003 edition of QA@CLEF revealed that the two most important sources of errors were answer selection and named entity recognition [10]. For this year's task, we used a new final answer selection module (similar to that described in [7]) with heuristic candidate answer filtering and merging and with stream voting, both to improve answer selection and to filter out NE errors.

3 Runs

We submitted two runs for the monolingual Dutch QA task—`uams041nlnl` and `uams042nlnl`—, and one run for the bilingual English to Dutch task—`uams041ennl`. All runs return exact answers, and combine answers from all streams. The `uams042nlnl` run is identical to `uams041nlnl`, except that it executes additional filtering and sanity checks on the candidate answers before final answer selection. These checks included zero-count filters (assuming that answers which do not appear as a phrase on the web are incorrect and that questions for which the focus does not appear in the local collection have no answer), and type-checking for location questions [16] (see Section 2.5). Our bilingual run included a simple translation of the questions from English to Dutch using a publicly-available interface of Systran (`http://www.systranet.com`), and then using Quartz-N for the translated questions.

4 Results and Further Analysis

Table 1 shows the evaluation results of our CLEF 2004 submissions. In addition to the number of right, wrong, inexact, and unsupported answers for all 200 questions, we

					Overall	*Accuracy*	*Accuracy*	*NIL accuracy*	
Run	*Right*	*Wrong*	*Inexact*	*Unsupp.*	*accuracy*	*over F*	*over D*	*precision*	*recall*
uams041nlnl	88	98	10	4	44.00%	42.37%	56.52%	0.00	0.00
uams042nlnl	91	97	10	2	45.50%	45.20%	47.83%	0.56	0.25
uams041ennl	70	122	7	1	35.00%	31.07%	65.22%	0.00	0.00

Table 1. Official results of our three submitted runs; the total number of test questions was 200. “Overall accuracy” is the percentage of questions answered correctly, “Accuracy over F (D)” is the percentage of factoid (definition) questions answered correctly, and “NIL accuracy” concerns the performance on questions with no known answer in the corpus.

also report accuracy figures (the percent of correct answers) for factoid and definition questions separately.

The run uams042nlnl scored slightly better than uams041nlnl. Interestingly, the gain is only in the factoids: uams042nlnl scored worse than uams041nlnl on definitions. Had we combined the answers to factoid questions produced by uams042nlnl with the answers to definition questions produced by uams041nlnl, we would have obtained on overall accuracy of 46.5%. This suggests that factoids benefit from additional checks and filters (which work well on short candidate answers), while definition questions benefit from a more lenient approach.

Additionally, our filters prove useful for detecting questions with no answers: 5 out of the 9 NIL answers returned (as part of the run uams042nlnl) were correctly identified using the filters, while none were identified without them.

When we compare the results of our Dutch QA system with the results of our participation in the QA track at TREC 2004 [1], we find that our Dutch QA system performs much better than our English QA system. It seems that the type of questions that are asked in the CLEF task are much easier for our Dutch QA system than the ones asked in the TREC task. One difference is that the questions at QA@CLEF are much shorter and additionally are back-generated from the CLEF corpus. In contrast, for the QA track at TREC the test questions are mainly compiled from log-files. Another probable explanation is that we spent more tuning our Dutch QA system than our English version.

4.1 Ranking Candidate Answer

Our system produces a ranked list of candidate answers, and then the highest ranked candidate is considered to be *the* answer to the question. Table 2 gives an evaluation over the ranking scheme: the number of correct answers at different cut-off levels and the Mean Reciprocal Rank (MRR). Our ranking method seems to be quite robust: only 12% of the questions are answered at ranks worse than 3, while for 65% one of the top-3 answers is correct. For 155 questions (77.5%) the system did extract a correct answer candidate (with an average of 26 candidates per question), and for 62% of these questions the correct answer was ranked highest.

Top n-answers	uams041nlnl	uams042nlnl	uams041ennl
top 1	96 (48%)	94 (47%)	70 (35%)
top 2	122 (61%)	117 (58%)	85 (42%)
top 3	131 (65%)	123 (61%)	96 (48%)
top 10	142 (71%)	133 (66%)	117 (58%)
top 20	152 (76%)	139 (69%)	122 (61%)
any rank	155 (77%)	141 (70%)	125 (62%)
MRR	0.57	0.55	0.43

Table 2. Evaluation of our ranking mechanism for the mono- and bilingual runs: the number of questions answered correctly and the mean reciprocal rank (MRR).

Note that the evaluation results presented differ somewhat from the official results in Table 1, because in our automatic evaluation, unsupported and inexact answers were also taken into account.

4.2 Contributions of the Streams

To analyze the contribution of different answer streams to the performance of the whole system, we carried out a number of experiments, disabling each stream individually and evaluating the resulting sub-systems using the assessors' judgements available for our official runs. The *Lookup* stream proved to be the most essential (the system answered 19 fewer questions when the *Lookup* was switched off), followed by the *Web Ngrams* stream (13 questions), *Collection Pattern Match* stream (4 questions) and *Collection Ngrams* (3 questions).

We also evaluated performance of each stream separately. Again, the *Lookup* stream had the best results, answering 57 questions (28.5%) on its own, while the precision-oriented *Pattern Match* streams answered the smallest number of questions (20 and 18, from the collection and Web, respectively).

As in our previous experiments, every stream does find a number of answers, but some streams seem more orthogonal to the rest of the system, answering questions that no other stream is capable of answering. Other streams are more redundant—for example, the *Quartz-E* stream itself found 20% of the answers, but the system had the same performance even without this stream. We should note that our final answer selection module makes use of the essential redundancy of the multi-stream architecture: 70% of the correct answers come from two or more answer streams!

We also compared two variants of the *Lookup* stream: an older version, which consults the databases extracted using only surface text patterns, and a new one, incorporating the results of the syntactic pattern extraction module on the dependency-parsed collection (similar to [8]). Although the tables from the syntactic module are much bigger, they also contain a significant amount of noise, which can potentially hurt the performance of the *Lookup* stream and the whole system. The version of the stream with the syntactic extraction module answered 8 questions more than the surface-based one. Evaluation of the two streams within the whole system showed that the syntactic extraction method helped Quartz to answer 2 more questions. This also supports the validity of our recall-based approach to QA: all possible ways to find answers should be exploited, and then the candidates should be carefully checked and cleaned.

4.3 Comparing the Mono- and Bilingual Runs

Since the questions for the bilingual task are translations of those in the monolingual task, it is interesting to compare the performance of the two types of runs. The overall accuracy of the bilingual run `uams041ennl` is lower than that of the monolingual runs, as was to be expected. The drop in accuracy can largely be attributed to the imperfect machine translation. Surprisingly, the correct answers in this run are not a subset of the correct answers found by the monolingual runs; while 44 questions (22%) were answered correctly by `uams041nlnl` and not by `uams041ennl`, there are 25 questions (12.5%) that were answered correctly by the bilingual run and not the monolingual one. Does translation make some questions easier to answer?

monolingual:	*Q3. Met hoeveel groeit de wereldbevolking elk jaar?*
	(With how much does the world's population grow each year?)
bilingual:	*Q3. Hoeveel verhoogt de wereldbevolking elk jaar?*
original question	(How much does the world population increase each year?)
monolingual:	*Q35. Waar is de Al Aqsa moskee?*
	(Where is the Al Aqsa moskee?)
bilingual:	*Q35. Waar is Al Moskee Aqsa?*
original question	(Where is the Al Aqsa Mosque?)
monolingual:	*Q25. Hoeveel jaar heeft Nelson Mandela in de gevangenis doorgebracht?*
	(How many years did Nelson Mandela spend in prison?)
bilingual:	*Q25. Hoeveel jaren van opsluiting diende Nelson Mandela?*
original question	(How many years of imprisonment did Nelson Mandela serve?)
monolingual:	*Q116. Hoe heet de premier van Rwanda?*
	(How is the premier of Rwanda called?)
bilingual:	*Q116. Wie is de Rwandese Eerste Minister?*
original question	(Who is the Rwandese Prime Minister?)

Table 3. A closer look at question translations.

We carefully analyzed the differences between bilingual and monolingual runs. There were five questions which were only answered by `uams041ennl` but not by the monolingual run. In all other cases, `uams041nlnl` did find the correct answer candidate, but it was not ranked highest. One reason why correct answers were found only in the bilingual run was that the questions were slightly reformulated and synonymous words were used (e.g., “verhoogt” instead of “groeit”—“grows,” in question *Q3*; see Table 3) or the word order was changed by the translation module (e.g., “Al Moskee Aqsa” instead of “Al Aqsa moskee” in question *Q35*). A different type of reformulation is illustrated by questions *Q25* and *Q116* in Table 3, where the sentences were changed more dramatically. An interesting point is that the reformulation (actually, double translation) does not necessarily result in grammatically correct sentences, but can

`uams041ennl`	answer correct	answer wrong	
question type correct	86	87	173
question type wrong	10	17	27
	96	104	200

Table 4. Accuracy vs. question classification for `uams041ennl`.

still lead to a useful paraphrasing. Apart from reformulation, we found that 6 questions in the bilingual run were identical to the ones of the monolingual run. However, in those cases, the answers of the bilingual run were ranked differently than in the monolingual run, leading to 6 correctly answered questions of the bilingual run.

4.4 Error Analysis

In this section we take a closer look at the errors made by our system, more specifically, at the errors made by our question classifier. For the run labeled `uams041nlnl` our system could not assign a question type to 9 questions (4.5%). In the bilingual run, `uams041ennl`, the number of questions without a question type increases to 24 (12%), which shows that our classifier is sensitive to lexical and grammatical features coded in the patterns; see Table 4. The evaluation of the question classifier based on the `uams041nlnl` run shows that in total, 27 questions were incorrectly classified (this includes the questions with no type assigned) and 10 of them were nonetheless correctly answered by the system. Out of the 87 incorrectly answered questions, 17 were misclassified which means that misclassification could have led to wrong answers in as many as 17 cases. The subsequent table displays the results of the evaluation of the question classifier.

The classifier could not assign a type to difficult questions like *Q167* and *Q168*:

Q167.	*Wat verkoopt Oracle?*
	(What does Oracle sell?)
Question type	none
Candidate 1	Microsoft
Q168.	*Wat bouwt Frank Gehry in Bilbao?*
	(What is Frank Gehry building in Bilbao?)
Question type	none
Candidate 1	Guggenheim Museum

Although our system did not assign a question type to question *Q168,* it did find the correct answer: the candidate co-occurring with the question terms happened to be the correct one. For a similarly difficult question Q167, is was not the case: as a competitor of Microsoft, Oracle often appears close to the word “Microsoft,” but the answer is of the wrong type. In most cases, where the question did not receive a question type, the type could only be derived from from the semantics of the verb and the question word.

Many instances of misclassification are either due to classification patterns that are mainly based on the question word, or to the fact that the arguments of the question are not correctly taken into account. The question type “manner” is assigned to question *Q44* as only the question word “hoe” is considered. In question *Q73*, “welk” and “president” lead to the question class “agent” whereas the correct class is “organization.” Lexical ambiguities are also a source of errors, like the word “positie (position)” which can occur in the context of a geographic location or in the context of a category of employment.

Q44.	*Hoe wordt de snelheid van een chip gemeten?*
	(How does one measure the speed of a chip?)
Question type	manner
Candidate 1	gemiddelde snelheid (average speed)
Q73.	*Van welk bedrijf is Christian Blanc president?*
	(Christian Blanc is president of which company?)
Question type	agent
Candidate 1	Morgen raad Hans (tomorrow committee Hans)
Q172.	*Welke positie had Redha Malek in 1994?*
	(Which position did Redha Malek have in 1994?)
Question type:	location
Candidate 1	Algerije (Algeria)

Our error analysis suggests that deeper features of the questions often need to be used by the classifier: verb semantics and intersections with question words, predicate-argument structure, etc. Moreover, a different expected answer type extraction strategy might be needed for bilingual QA, where translated questions are often not well-formed sentences.

5 Discussion and Future Work

We presented our multi-stream question answering system as well as the official runs it produced for CLEF 2004. Running in parallel several subsystems that approach the QA task from different angles proved successful, as some approaches seem better suited to answering certain types of questions than others. Although this year’s task was made more complex through the inclusion of definition questions, we were able to slightly increase the performance of our system. It seems that the combination of improving modules and incorporating additional information sources (such as the Dutch Wikipedia) led to the reported improvements. We found that some of the correct answers found by the bilingual run were not amongst the correct answers of the monolingual run. This suggests that the translation procedure produces paraphrased questions—grammatical or not—which in turn yield different answers. Thus, a combination of the two tasks will likely increase the recall of our system, and, with a careful answer selection procedure, this might lead to higher overall accuracy scores.

We also found that our system performs much better on Dutch at QA@CLEF than on English questions at the QA track at TREC 2004. One obvious reason was that the Dutch version of our QA system can deal much better with back-generated questions that are based on the corpus from against which the questions have to be answered. Another reason might be that the patterns used in our Dutch QA system were written by native speakers and are more advanced than the ones for English. Finally, we simply spent more time fine-tuning and debugging our Dutch QA system than our English language version.

Our ongoing work on the system is focused on additional filtering and type checking mechanisms, and on exploiting high-quality external resources such as the CIA world fact book, Wikipedia, and WordNet. Our comparison of the monolingual and bilingual runs suggests that question paraphrasing through translation can be a useful method for improving recall. We are also working on refining and improving the closely related modules for question classification, named entity extraction and type checking, to address a frequent source of errors: the mismatch between the expected answer type and the answers found.

Acknowledgments

We are grateful to Gertjan van Noord for supplying us with a dependency parsed version of the Dutch CLEF corpus. This research was supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. In addition, Maarten de Rijke was also supported by grants from NWO, under project numbers 365-20-005, 612.069.006, 612.000.106, 612.000.207, 612.066.302, and 264-70-050.

References

[1] D. Ahn, V. Jijkoun, J. Kamps, G. Mishne, K. Müller, M. de Rijke, and S. Schlobach. The University of Amsterdam at TREC 2004. In *TREC 2004 Conference Notebook*, Gaithersburg, Maryland USA, 2004.

[2] G. Bouma, G. Van Noord, and R. Malouf. Alpino: Wide-coverage computational analysis of Dutch. In *Computational Linguistics in The Netherlands 2000*. 2001.

[3] T. Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000*, 2000.

[4] 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'02*, pages 291–298, 2002.

[5] M. Banko et al. AskMSR: Question answering using the Worldwide Web. In *Proceedings EMNLP 2002*, 2002.

[6] C. Foeldesi, K. Müller, and M. de Rijke. Using the Corpus Gesproken Nederlands to Build a Named Entity Recognizer. In *14th Meeting of Computational Linguistics in the Netherlands (CLIN-2003)*, 2003.

[7] V. Jijkoun and M. de Rijke. Answer selection in a multi-stream open domain question answering system. In S. McDonald and J. Tait, editors, *Proceedings 26th European Conference on Information Retrieval (ECIR'04),*, volume 2997 of *LNCS*, pages 99–111. Springer, 2004.

[8] V. Jijkoun, M. de Rijke, and J. Mur. Information extraction for question answering: Improving recall through syntactic patterns. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, 2004.

[9] V. Jijkoun, G. Mishne, and M. de Rijke. Preprocessing Documents to Answer Dutch Questions. In *Proceedings of the 15th Belgian-Dutch Conference on Artificial Intelligence (BNAIC'03)*, 2003.

[10] V. Jijkoun, G. Mishne, and M. de Rijke. How frogs built the Berlin Wall. In *Proceedings CLEF 2003*, LNCS. Springer, 2004.

[11] V. Jijkoun, G. Mishne, C. Monz, M. de Rijke, S. Schlobach, and O. Tsur. The University of Amsterdam at the TREC 2003 Question Answering Track. In *Proceedings TREC 2003*, pages 586–593, 2004.

[12] D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu, and O. Bolohan. LCC Tools for Question Answering. In E.M. Voorhees and D.K. Harman, editors, *The Tenth Text REtrieval Conference (TREC 2002)*. National Institute for Standards and Technology. NIST Special Publication 500-251, 2003.

[13] C. Monz and M. de Rijke. Tequesta: The University of Amsterdam's textual question answering system. In E.M. Voorhees and D.K. Harman, editors, *The Tenth Text REtrieval Conference (TREC 2001)*, pages 519–528. National Institute for Standards and Technology. NIST Special Publication 500-250, 2002.

[14] C. Monz and M. de Rijke. Shallow morphological analysis in monolingual information retrieval for Dutch, German and Italian. In *Proceedings CLEF 2001*, LNCS. Springer, 2002.

[15] N. Oostdijk. The Spoken Dutch Corpus: Overview and first evaluation. In *Proceedings LREC 2000*, pages 887–894, 2000.

[16] S. Schlobach, M. Olsthoorn, and M. de Rijke. Type checking in open-domain question answering. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, 2004.