Downloaded from UvA-DARE, the institutional repository of the University of Amsterdam (UvA)
http://hdl.handle.net/11245/2.38040

File ID	uvapub:38040
Filename	uams-trec-2004-final-qa.pdf
Version	unknown

SOURCE (OR PART OF THE FOLLOWING SOURCE):

Type	conference contribution
Title	Using Wikipedia at the TREC QA Track
Author	D.D. Ahn
Faculty	FNWI: Informatics Institute (II)
Year	2005

FULL BIBLIOGRAPHIC DETAILS:
http://hdl.handle.net/11245/1.241824

Copyright

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use.

UvA-DARE is a service provided by the library of the University of Amsterdam (http://dare.uva.nl)
(pagedate: 2014-08-16)

Using Wikipedia at the TREC QA Track

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

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
http://ilps.science.uva.nl/

Abstract: We describe our participation in the TREC 2004 Question Answering track. We provide a detailed account of the ideas underlying our approach to the QA task, especially to the so-called "other" questions. This year we made essential use of Wikipedia, the free online encyclopedia, both as a source of answers to factoid questions and as an importance model to help us identify material to be returned in response to "other" questions.

1 Introduction

In this paper we describe our participation in the TREC 2004 Question Answering track; our participation in the Web and Terabyte tracks is described elsewhere [8]. This year, we had several aims for the Question Answering track. One was to extend our QA system to handle this year's more complex question presentation, and to see how our existing modules cope with this new setting. Another was to make good use of the English edition of Wikipedia (http://en.wikipedia.org), the open domain encyclopedia, both as an additional stream for answering factoid questions, and as an *importance model* to help us answer "other" questions. Let's explain the latter point before we continue.

When the QA track at TREC was introduced, it focused on so-called "factoid" questions (typically having a short named entity as an answer) such as *How many people live in Tokyo?* or *When is the Tulip Festival in Michigan?*. As the track evolved, it was argued that this type of questions does not accurately model the needs of real users of QA technology. In addition to named entities as answers, users often search for definitions of concepts, or for summaries of important information about them. As a result, in 2003 TREC introduced *definition questions*—questions for which the answer is not a single named entity, but a list of *information nuggets* [14].

We were glad to see that at the TREC 2004 QA track this was taken a step further. The questions were now clustered in small groups, organized around the same topic. For example, the topic *Concorde* included questions such as *How many seats are in the cabin of a Concorde?* and *What airlines have Concordes in their fleets?*. Finally, for every topic, the track guidelines required participants to supply "additional important information found in the corpus about the target, that was not explicitly asked." This last requirement has been dubbed *"other"* questions. In our view, the task presented at the TREC 2004 QA track, and the introduction of the "other" questions makes a big step towards more realistic user scenarios. According to our own analysis of web query logs, users tend to ask much more "knowledge gathering" questions than factoid questions about specific facts.

This new type of "other" questions puts more emphasis on the *user* aspect in the QA process—an issue that has mostly been neglected in the QA community. The TREC criteria for what is a *good answer* to a given question has so far been rather vague, but QA systems dealt with this vagueness fairly effectively for factoid questions. With the "other" questions, where systems are required to re-

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

turn only *important* information, there is an implicitly assumed user model that can discriminate between important and unimportant facts about a topic. For example, for the topic *Clinton*, his birthday might be considered important, while the day of the week when he left Mexico probably is not. In order to give reasonable responses to "other" questions, a QA system needs to model such preferences.

We present an approach for answering "other" questions using an explicit "importance" model. We describe a method for gathering important facts about an entity from a collection of documents and for ranking the facts with respect to their importance for the user. We show that our ranking improves over plain retrieval of facts from the corpus. The core idea of our method is to estimate the importance of facts found in the target collection by using external "reference" corpora, high-quality sources of information that model a user's ability to distinguish between important and unimportant facts. The proposed method is our first step towards user-oriented QA, and further refinements of the underlying techniques are needed. We identify additional areas where this method is or may be helpful, and discuss its strengths, weaknesses and directions for further research.

The rest of this paper is organized as follows. We give separate accounts of our approaches for answering factoid/list questions and for answering "other" questions. First, though, we address a complication in the presentation of questions in this year's QA task: the grouping of questions by *target*. We then describe our runs, present our results, and conclude.

2 Handling Targets

Each target is given explicitly as a phrase, and the questions for the target are presented in sequence. The possibility of anaphoric dependencies of the questions on the target or on preceding questions is thus introduced. We use an anaphoric resolution module to resolve pronouns occurring in the questions. Our module is simple: each pronoun is resolved to the highest ranked compatible antecedent in the antecedent list. The antecedent list consists of the target and all noun chunks occurring in preceding questions (with previously resolved pronouns replaced by their antecedents).

Our heuristic is to rank the target highest and to rank the other noun chunks according to their occurrence order. Compatibility is determined according to a simple type system: pronouns are marked **human** (*he*, *she*, etc.), **non-human** (*it*, *this*, *that*, etc.), or **unknown** (*they*, etc.), while other noun chunks are unmarked until they are resolved to a pronoun. These simple heuristics appear to work well with the question groups, where few entities are introduced and where there is a strong tendency to refer to the target.

3 Factoid Questions

Our approach for answering factoid questions is largely based on QUARTZ, our QA system used for experiments in the TREC 2003 QA track [7] and the CLEF 2004 Question Answering track [6]. We use an architecture where several streams run in parallel: each is based on a different approach to QA and is a self contained QA system in itself. A final step of merging the results of the streams is based on both redundancy of answers between streams and a process of learning the strengths and weaknesses of each of them [3].

This year, apart from minor technical modifications of these streams, we employed two new components in QUARTZ: an additional stream exploiting an open-domain encyclopedia and a mechanism for type checking of the answer candidates generated by each stream; see Figure 1. We also implemented a number of simple filtering mechanisms that serve as sanity checks for the answer candidates and performed a number of experiments in our answer justification module. We give an overview of the new components here and refer the reader to [3, 4, 6, 7] for an account of the rest of the system.

3.1 Encyclopedia Stream

Many systems participating in the TREC QA track use not only the local (AQUAINT) corpus, but also additional knowledge sources such as the web and various gazetteers [15]. The use of external resources (such as the web) in QUARTZ has proved to be beneficial, and we have therefore decided to employ an additional source of external knowledge into the system: a corpus specifically designed to address the information needs expressed by

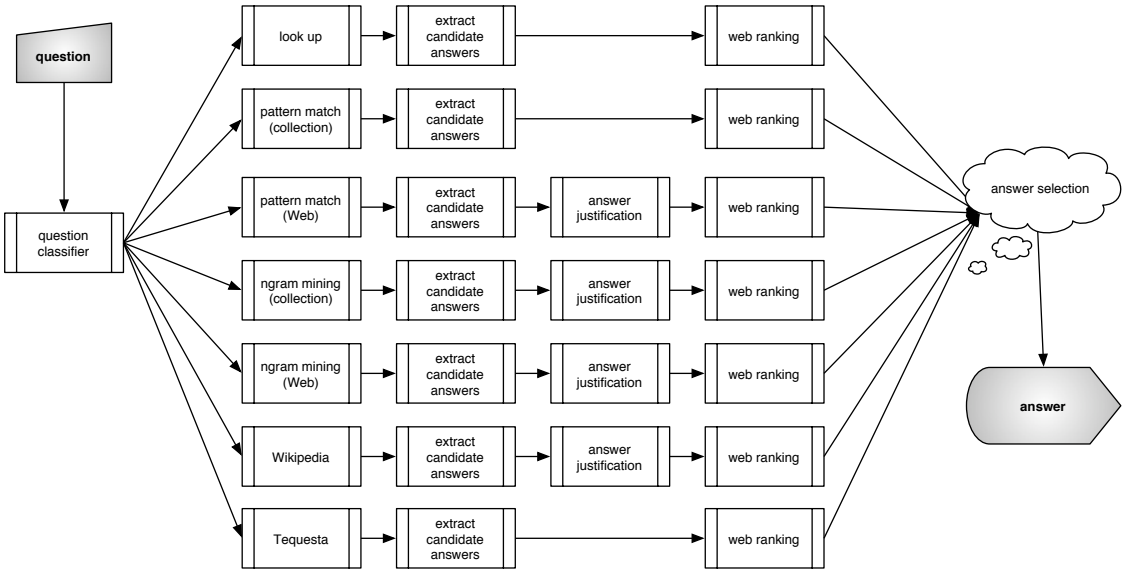

Figure 1: QUARTZ System Overview.

the open-domain questions appearing in TREC—an encyclopedia.

We used the English edition of Wikipedia (`http://en.wikipedia.org`), a free-content encyclopedia: among the reasons to use it are its relatively wide coverage, its availability in a standard database format, and the fairly structured format of its entries.

We adopt a simplification of techniques reported already in [9]. Given a question and the question topic, we first extract the Wikipedia entry for the topic. The QUARTZ question classifier module identifies the named entity type that should be returned as an answer to the question; a named entity tagger then identifies potential answers in the encyclopedic entry. The list of answers is then ranked according to two factors: the *prior answer confidence*, which is an estimate of how likely it is that the named entity is an answer to any question, and a *posterior answer confidence*, which is an estimate of the likelihood of the named entity to be an answer to the question at hand. For estimating the prior confidence, we use layout information about the Wikipedia format, basically giving more confidence to named entities appearing earlier in the entry; for the posterior estimations, we calculate a sentence-level similarity score between the question and sentence containing the answer, based on the Jaccard measure. The final ranking of the answers is a combination of the prior and posterior estimations, with more weight given to the posterior one.

3.2 Answer Type Checking

In question analysis, search, and extraction of answer candidates, QUARTZ, like other QA systems, applies a recall oriented strategy. The underlying assumption is that recall can be maintained at an acceptable level in the early steps of the QA process because possible noise will be filtered out in the final filtering step.

Answer type checking—checking whether a given answer candidate belongs to the expected semantic type (or set of types)—is one filtering method that we explored further in this year's TREC evaluation. The factoid questions used in the TREC QA track are associated with a small number of semantic types—the expected types of the correct answers. On top of the coarse-grained expected answer types used to extract answer candidates (such as PERSON, LOCATION, DATE or ORGANIZATION), we found it useful to identify more precisely whether we are looking for, e.g., an ACTOR, a CAPITAL,

a YEAR, or an NGO (Non-Governmental Organization).

We extended previous experiments in domain-specific type checking [13] to an open-domain type checker by combining two fitering approaches, *ontology-based* and *redundancy-based*. First, we extract a WordNet synset as the required expected answer type of a question. For each candidate answer, we then calculate the probability that it has an expected type, based on word co-occurrence of the answer and the expected type on the web. Finally, an answer is filtered unless it is more likely to be of the expected answer type than of one of its WordNet siblings. Due to space restrictions, a more detailed description and a formal evaluation of this new type checker will be published elsewhere.

3.3 Additional Filtering

In previous evaluations, we have encountered the problem of "junk"—ungrammatical answers resulting mainly from the combination of different streams and the heavy usage of n-gram techniques in QUARTZ [5]. This year we employ a simple *web hit count filter* to cope with this phenomenon. Each candidate answer is sent, as a phrase search, to Google, and phrases that have no results at all are considered to be incorrectly formed answers and removed from the candidate list.

Additionally, since our system also relies on external knowledge such as the web and Wikipedia, it often obtains answers even for questions with no answer in the collection (NIL questions). We attemp to detect such questions using another simple filter, a *collection hit count filter*: each question topic is searched for in the collection and, if no documents are retrieved, a NIL response is given for the question.

3.4 Answer Justification

An additional substantial problem found in previous evaluations is that of *unsupported answers*, i.e., correct answers with an incorrect supporting document. We have invested some effort in improving our answer justification mechanism, with an improvement of more than 20% on training data; even so, unsupported answers still account for half of our total number of correct answers.

Previously, we used Okapi-based retrieval for answer projection, with the query formed from the question and the answer. The Okapi model's good performance on early precision allowed us to take the top retrieved document as the supporting document. For this year, we still base our projection mechanism on retrieval only, but have moved from Okapi to a vector space model with extensive usage of various query operators in the query. We issue the answer as a phrase term, identify phrases in the question and issue them as phrase terms as well, and use boolean operators for various terms in the query. These are techniques that are known to increase early precision, and, as mentioned, we have indeed noticed an improvement on the training data.

4 List Questions

As in the previous TREC QA track, we have not implemented a specific mechanism to handle list questions, but rather used our factoid approach for these questions, as well. The top ranking answers according to this approach are given as the answer to the list question; the number of answers depends on a confidence drop in the scores assigned to the candidates; in the absence of such a drop, a fixed threshold is used.

5 Answering "Other" Questions with an Importance Model

In this section we provide the details of our method for extracting, ranking, and re-ranking information nuggets from a corpus. In a nutshell, after identifying a suitable "reference" corpus for our domain (our user model), we first use IR and NLP methods to identify information nuggets—short excerpts of text—related to the topic, both from the given document collection and from the "reference" corpus. Then, we use sentence-similarity metrics to rank the nuggets from the collection: the facts similar to those found in the "reference corpus" are considered more important and ranked higher.

5.1 Target Corpus and Reference Corpus

In the TREC QA task, answers to questions (including "other" questions) must be found in a given text corpus. In recent years, this corpus has been a part of the AQUAINT

corpus, containing more than 1 million newswire documents, and a total of 3.1GB of text. In our experiments this corpus is used as the *target* corpus, where important information nuggets have to be located. The corpus is unstructured: we do not know beforehand which articles or passages contain "important" information about a topic.

The "reference" corpus to be used should be a relatively small, high-quality collection of documents, which is catalogued in a way that facilitates selecting documents which contain important information for a given topic. Typical corpora that can be used for such reference purposes are encyclopedias (e.g., biography pages from `http://biography.com`) and various knowledge bases (e.g., the Internet Movie Database `http://www.imdb.com`). Since TREC QA is an open domain task, we used the English edition of Wikipedia (`http://en.wikipedia.org`), an open domain encyclopedia. The version we used contained 768,000 entries (including placeholders and disambiguation entries), for a total of 900 MB of text.

5.2 Mining Facts from the Target Corpus

When answering an "other" question for a given topic, we use IR to locate documents containing information about the topic, and then split the sentences from the retrieved documents into more easily "digestable" shorter nuggets.

Retrieval

First, from the target collection we retrieve the top 20 documents containing the topic as a phrase, using a traditional vector space model for the retrieval. Our collection is composed of news articles with headlines. Since an occurrence of a topic in a headline can be very indicative of the document's importance for the topic, we indexed the headlines and the article bodies separately, and calculate the retrieval score as a combination of the different representations; this is a common technique for semi-structured IR [12].

Extraction

Since a response to an "other" question is a list of short nuggets, we have to split the retrieved documents into separate facts. This raises several problems. First, we observed a notorious use of referential NPs: even in highly focused documents the topic is introduced initially, and then referred to with pronouns or definite NPs (e.g., "*PRESIDENT CLINTON arrived today at the . . . HE will leave to Mexico on Monday*"). We therefore resolve pronouns in the documents using the simple anaphora resolution module described in Section 2. Then, we extract all sentences which contain the topic (either originally or after the resolution); this is a natural way to restrict our attention to document sections which potentially include facts about the entity.

Still, the sentences are often too long to be presented as nuggets. Moreover, as the next step of our method involves comparison of nuggets, we need to keep them atomic, i.e., as short as possible. We observed that most facts in the extracted sentences could be described with simple predicates (e.g. "[President Clinton] *will leave* to Mexico"). We therefore parse the sentences with Minipar—a wide-coverage dependency parser [10]—and consider as a *fact nugget* every predicate (usually, a verb) with all its arguments and modifiers. Table 1(a) shows an example for the topic *Cassini space probe*.

Finally, every extracted fact is given a *prior importance estimation*: the retrieval score of the document from which the fact was extracted.

5.3 Mining Facts from the Reference Corpus

In order to obtain a list of "good" facts for a given topic, we now repeat the fact extraction stage, with slight modifications, for the reference corpus. First, we extract a high-quality document (i.e., an encyclopedia entry) for the topic. We then apply the anaphora resolution and sentence splitting methods described in the previous section. Next, we assign *importance* to each fact, based on layout cues in the document, such as proximity to the beginning of the entry. These heuristics are based on the fact that in encyclopedia entries, important information is typically given first, data in tables is usually significant, and so on. An example of facts extracted from an encyclopedia entry is given in Table 1(b).

Document text	The Cassini space probe , due to be launched from Cape Canaveral in Florida of the United States at dawn , is carrying 33 kg of plutonium needed to power A rocket's seven-year journey to Venus and Saturn . Local mass media quoted opponents of Cassini as saying at the weekend that the mission will cross Panama , the Caribbean , Southern Africa and Madagascar be fore hurtling into space . Foreign affairs spokesman Pieter Swanepoel said neither had anything's department received any request or contacted the American authorities to find out what was happening with Cassini . . . .
Extracted facts	• The Cassini space probe : due to be launched from Cape Canaveral in Florida of the United States at dawn • Local mass media quoted opponents of Cassini as saying at the weekend the mission will cross Panama • Foreign affairs spokesman Pieter Swanepoel said neither had anything's • department to find out what was happening with Cassini • . . .

(a) Extracting facts from the target corpus.

Encyclopedia entry	Cassini-Huygens is a joint NASA/ESA unmanned space mission intended to study Saturn and its moons. The spacecraft consists of two main elements: the Cassini orbiter and the Huygens probe. The spacecraft was launched on October 15 , 1997 and entered Saturn's orbit on July 1 , 2004. October 15, is the first spacecraft to orbit Saturn and just the fourth spacecraft to visit Saturn.
Extracted facts	1. Cassini - Huygens a joint NASA/ESA unmanned space mission intended to study Saturn and its moons 2. The spacecraft consists of two main elements 3. the Cassini orbiter the Huygens probe 4. The spacecraft entered Saturn's orbit on July 1, 2004 5. October 15, is the first spacecraft to orbit Saturn just the fourth spacecraft to visit Saturn

(b) Extracting facts from the reference corpus.

Re-ranked facts	• Cassini will be carrying 12 separate packages of scientific instruments a probe [3] • Saturn's largest moon [1] • department to find out what was happening with Cassini [3] • the instruments on Cassini to provide pictures of Saturn Nearly seven meters' rings moons radar to pierce the orange [1] • . . .

(c) Re-ranked facts from the target corpus (with the id of the most similar reference fact in brackets).

Table 1: Fact extraction and re-ranking in action.

5.4 Estimating Importance of Facts

At this stage, we have two lists of nuggets: facts from the target corpus, with prior importance estimation, and reliable facts from the reference corpus, each with its importance value. To refine the importance estimation for the target facts, we calculate sentence-level similarity between the target and reference nugget lists: we exhaustively compare each target fact to each fact from the reference corpus. We experimented with two types of sentence-level similarity measures: lexical and semantic.

We measure lexical similarity by determining the word overlap between the sentences, using metrics such as Jaccard [2] to normalize over the sentence lengths. Prior to the comparison we use standard stemming and stopword removal on both sentences to increase the morphological uniformity. As to semantic similarity between sentences, we use linguistically motivated techniques to find similarities also between sentences which do not match on the surface level. We use two types of metrics; the first is the total WordNet distance of words appearing in the sentences, based on methods described in [1]. Alternatively, we use similarity scores between pairs of words derived from proximities and co-occurrence in large corpora, described in [11], and sum the total proximity measure for the words in the two segments.

In the experiments described below we used the lexical similarity with Jaccard metric. Later we found that co-occurrence-based measures seem to give better estimates of sentence similarity. A careful evaluation of different measures for this task is in our future plans.

Let $\{t_i\}$ denote the list of facts extracted from the target corpus and $\{r_j\}$ the reliable facts from the reference corpus. We denote the similarity between two facts as $\mathrm{sim}(t_i, r_j)$, the prior importance estimation of a target fact as $I_{pr}(t_i)$ and the importance of the reliable fact as $I(r_j)$. Then the updated, posterior importance estimation of a target fact is calculated as follows:

$$I_{post}(t_i) = I_{pr}(t_i) \cdot \max_j \left(I(r_j) \cdot \mathrm{sim}(t_i, r_j) \right).$$

We sort the target facts by decreasing posterior importance and present the top N as the key facts about the topic.

5.5 Removing Redundant Facts

At the TREC 2004 QA track, each "other" question was asked after a sequence of factoid questions, all about a given topic. Therefore, an additional requirement was set on the response to the "other" question: the retrieved facts should not duplicate the information conveyed by (answers to) the factoid questions.

To avoid such duplication, we performed another filtering step: from the ranked list, we omit nuggets that are *similar* to other nuggets higher in the ranking, or to one of the factoid questions together with its answer (as found by our factoid-QA system). We use the same sentence-level similarity measure $\mathrm{sim}(\cdot,\cdot)$ as for the posterior importance estimation.

6 Runs

We submitted 3 runs differing only in the final sanity checking for answer candidates. Our aim here was to compare different options for the answer filtering.

`uams04raw` No answer type checking or other filtering mechanisms employed.

`uams04tc1` Answer type checking and web and collection hit-count filters described above used.

`uams04tc2` Same as previous run, but Wikipedia not used for the 'other' questions.

7 Results

Table 2 gives the combined results for the 3 QA tasks (accuracy for factoids, F score for list and "other" questions) and the final scores of our runs.

Table 2: Results for the QA track

Run identifier	A (Exact,Lenient)	F (List)	F (Oth)	Overall
`uams04raw`	0.135 , 0.287	0.094	0.210	0.143
`uams04tc1`	0.126 , 0.269	0.085	0.207	0.136
`uams04tc2`	0.126 , 0.269	0.087	0.184	0.131

The results are disappointing, especially in light of our recent good performance for Dutch Question Answering [6] (where the questions are easier than the TREC

questions, and rather similar to the TREC8 or TREC9 QA track). An error analysis shows that most errors are due to the insufficiently fine-grained question classification and entity extraction (the current system uses only 37 question types and 5 NE types). Also, we note the high rate of unsupported answers: without the answer justification requirement, our performance would double. Furthermore, our type-checking module does not seem robust enough yet to improve the performance of the system.

Let's take a closer look at our performance on the "other" questions. We compare two runs, a baseline run (`uams04tc2`) and a re-ranked run (`uams04raw`). In the former we extracted nuggets from the target collection as described above, and used prior estimates of the facts to rank the nuggets; 20 or fewer nuggets were submitted. For the latter, we used posterior estimates of the nugget importance (I_{post}) instead; also 20 or less facts were submitted per topic. The results of the runs are given in Table 3. For comparison, the best system at TREC 2004 achieved an F-measure of 0.46, while the median F-measure over all 63 submitted runs is 0.184.

Table 3: Evaluation results for "other" questions.

Measure	Baseline	Re-ranked
Precision	0.176	0.220 (+25%)
Recall	0.208	0.237 (+14%)
F-measure	0.184	0.210 (+14%)

Note that the version of the F-measure used at TREC 2004 is biased towards recall. As is clear from Table 3, our re-ranking method substantially improves both recall and precision, but more so for precision.

A further per question breakdown of the change of performance in terms of F-measure is given in Figure 2 (top), indicating that while the gain in F-measure averaged over all questions is positive, there are questions whose score is affected negatively by our re-ranking mechanism. The results in Table 3 indicate that our re-ranking mechanism affects precision and recall differently; this is reflected in Figure 2 (middle) and (bottom), where we provide per-question breakdowns for precision and recall. For 26 questions the F-measure of the original sentences (without re-ranking) is 0, preventing any improvement from our re-ranking method.

An analysis of the assessed runs revealed that often

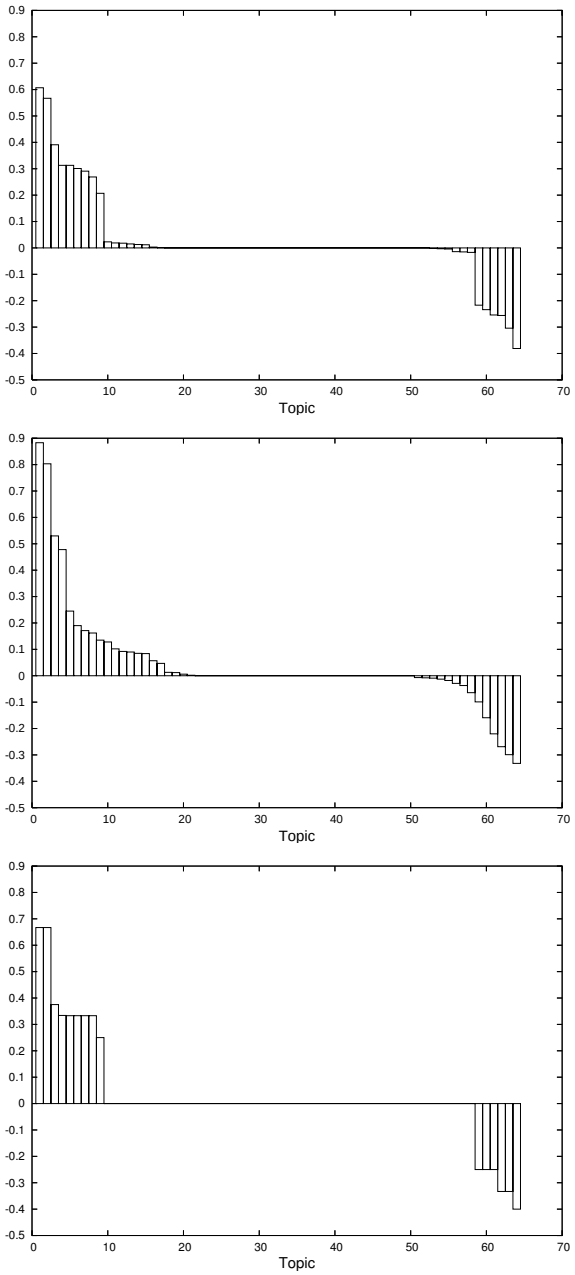

Figure 2: Per-question breakdown of effect of re-ranking on "other" questions: F-measure (top), precision (middle), and recall (bottom).

good nuggets were in the collection, but not in the top 20 documents we used for the extraction. Indeed, the threshold of 20 was set mainly for computational reasons, and further experiments with higher thresholds has shown clear improvements. Since the evaluation of new runs has to be done manually, we have no numerical support for this claim. Another major source of errors was the similarity measure (normalized word overlap) used in our submitted runs. Because of its sparsity, often the decision about a match was based on a single common word, as, e.g., for the third nugget in Table 1(c). Again, experimenting with more suitable measures is hindered by the lack of automatic evaluation methodology: unlike the factoid questions at TREC, for "other" questions it is difficult to create patterns of correct answers. Developing such effective automatic evaluation methods is essential for improving the systems. Re-ranking errors were also caused by our sentence splitting and anaphora resolution methods. For example, for the topic "*Carlos the Jackal*" one of the important nuggets "*the man known as Carlos the Jackal, once considered the world's most wanted terrorist, is serving a life sentence there*" was discarded after re-ranking, although the reference corpus provided the nugget "*on December 23 he was found guilty and sentenced to life imprisonment*." Although both contain the key words *sentence* and *imprisonment*, the nugget from the target collection was too long for the similarity to be detected by our method. A better sentence splitter (capable of ignoring reduced relative clauses) could make nuggets shorter and the similarity more obvious. Another reason for discarding the snippet was the incorrectly resolved referential "*there*." Had it been resolved to its true antecedent "*La Sante*," the snippet could have matched the reference nugget "*he was sent to La Santé de Paris prison to await trial*."

8 Conclusions

In this paper we have described our participation in the TREC 2004 Question Answering track.

This year, our work for the Question Answering track was largely motivated by the wish to extend our QA system to handle the new, more complex question presentation, and to exploit Wikipedia at various stages of our QA architecture, for factoids and for "other" questions. While the new setting proved tractable, a variety of bugs in the "core" of our QA engine lead to rather disappointing scores on the factoids.

As to the "other" questions, by comparing facts extracted from a target collection to the information from a reference resource (Wikipedia), we identified those facts that are potentially *important* for the user. Even with a simple word overlap-based similarity measure, this method shows reasonable performance: applying it to answer "other" questions in the TREC 2004 QA track, we show substantial improvements over the baseline. Our analysis of the TREC 2004 results on "other" questions suggests experimenting with more sophisticated sentence-level similarity measures and improving sentence splitting for extraction of atomic facts.

Acknowledgments

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 612-13-001, 612.000.106, 612.000.207, 612.066.302, and 612.069.006.

References

[1] M. D. Boni and S. Manandhar. The use of Sentence Similarity as a Semantic Relevance Metric for Question Answering. In *Proceedings of the AAAI Symposium on New Directions in Question Answering*, 2003.

[2] P. Jaccard. The distribution of the flora of the alpine zone. *New Phytologist*, 11:37–50, 1912.

[3] 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.

[4] V. Jijkoun, M. de Rijke, and J. Mur. Information extraction for question answering: Improving recall through syntactic patterns. In *Proceedings of*

the 20th International on Computational Linguistics (COLING 2004), 2004.

[5] V. Jijkoun, G. Mishne, and M. de Rijke. How frogs built the Berlin Wall. In *Proceedings CLEF2003*, volume LNCS. Springer, 2004.

[6] V. Jijkoun, G. Mishne, M. de Rijke, S. Schlobach, D. Ahn, and K. Müller. The university of amsterdam at qa@clef 2004. In C. Peters and F. Borri, editors, *Working Notes for the CLEF 2004 Workshop*, pages 321–324, 2004.

[7] 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.

[8] J. Kamps, G. Mishne, and M. de Rijke. Language models for searching in web corpora. In *This volume*, 2005.

[9] J. Kupiec. Murax: a robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 181–190. ACM Press, 1993. ISBN 0-89791-605-0.

[10] D. Lin. PRINCIPAR – an efficient, broad-coverage, principle-based parser. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, 1994.

[11] D. Lin and P. Pantel. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[12] P. Ogilvie and J. Callan. Combining document representations for known-item search. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM Press, 2003.

[13] 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)*, pages 398–402. IOS Press, 2004.

[14] E. Voorhees. Overview of the TREC 2003 question answering track. In *Proceedings Twelfth Text Retrieval Conference (TREC 2003)*, pages 54–68, 2003.

[15] E. Voorhees. Overview of the trec 2003 question answering track. In *The Twelfth Text REtrieval Conference (TREC-03)*, 2004.