The University of Amsterdam at the TREC 2007 QA Track

Katja Hofmann Valentin Jijkoun Mahboob Alam Khalid
Joris van Rantwijk Erik Tjong Kim Sang

ISLA, University of Amsterdam
http://ilps.science.uva.nl/

Abstract: In our participation in the TREC 2007 Question Answering (QA) track, we focused on three tasks. First, we processed the new blog corpus and converted it to formats which could be used by our QA system. Second, we rewrote the module interface code in Java in order to improve the maintainability of the system. And third, we added a new table stream which has learned associations between question properties and properties of candidate answers. In the three runs we submitted to the competition, we experimented with answer type checking and web re-ranking. In follow-up experiments we were able to further evaluate the contribution of these two factors, and to evaluate our new table lookup stream and combinations of streams.

1. INTRODUCTION

We participated in the TREC 2007 Question Answering (QA) track with the experiences of three earlier participations (2003-2005). Our prime goal this year was to evaluate a new QA stream, and to experiment with answer type checking, web re-ranking, and with combining QA streams.

From a system point of view, we continued with our efforts to approach QA as an XML retrieval task [1]. Since our last participation, we have standardized data access in the QA system by converting *all* of our data resources to fit in an XML database [5]. For the current evaluation, we have focused on three tasks:

1. Convert the new blog corpus to formats that can be used by our system.
2. Standardize the code that takes care of the communication between the different modules of the system by completely rewriting it in Java (previously it was written in Perl).
3. Include in the parallel architecture a new table stream which has *learned* associations between questions and candidate answers rather than rely on manually defined association rules.

This paper contains eight sections. After this introduction, we present a general overview of the system in section 2. Our work on processing the blog corpus is discussed in section 3 while section 4 describes our approach of QA as XML retrieval in more detail. Sections 5 and 6 present the two most recent modifications to the QA system: standardizing the code and adding a new table stream. Section 7 describes the submitted runs and discusses their performance. We conclude in section 8.

2. SYSTEM DESCRIPTION

The architecture of our Quartz QA system is an expanded version of a standard QA architecture consisting of parts dealing with question analysis, information retrieval, answer extraction, and answer post-processing (clustering, ranking, and selection). The Quartz architecture consists of multiple answer extraction modules, or *streams*, which share common question and answer processing components. The answer extraction streams can be divided into three groups based on the corpus that they employ: the newspaper corpus, the new blog corpus, or the Web. Below, we describe these briefly.

The Quartz system (Figure 1) contains four streams that generate answers from the two data sources, the AQUAINT newspaper corpus and the blog corpus. The *Table Lookup* stream searches for answers in specialized knowledge bases which are extracted from the corpus offline (prior to question time) by predefined rules. These information extraction rules take advantage of the fact that certain answer types, such as birthdays, are typically expressed in one of a small set of easily identifiable patterns. The stream uses the analysis of a question and manually defined patterns to map questions to database queries. Our new stream, *ML Table Lookup*, performs the answer lookup task by using a mapping learned automatically from a set of training questions (see section 6 for a more elaborate description). The *NGram* stream extracts the word ngrams that most frequently occur with words from the question. First, passages are retrieved from the collection using a standard retrieval engine (Lucene) and a text query generated from the question. Second, the most frequent ngrams in these passages are generated and filtered and returned as answers.

The most advanced of the four streams is XQuesta. For a given question, it generates XPath queries for answer extraction, and executes them on an XML version of the corpus which contains both text and additional annotations. The annotations include information about part-of-speech, syntactic chunks, named entities, and temporal expressions. For each question, XQuesta only examines text passages relevant to the question (as identified by Lucene, see also section 4).

There is one stream which employs textual data outside the TREC document collection defined for the task: the *NGram* stream also retrieves answers by submitting automatically generated web queries to the Google web search engine and collecting most common ngrams from the returned snippets. The answer candidates found by this stream are not backed up by documents from the TREC collection as required by the task. For this reason such candidates are never returned as actual answers, but only used at the answer merging stage to adjust the ranking of answers that are found by other QA streams as well.

3. PRE-PROCESSING THE BLOG CORPUS

In this year's TREC QA track the WEB_BLOG_06 collection (blog corpus) was introduced as an additional source of information. This

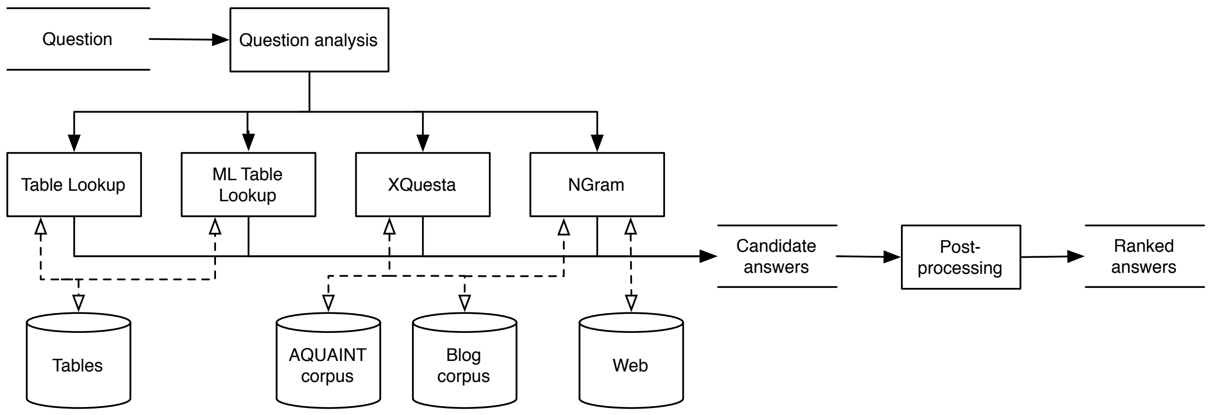

Figure 1: Quartz-2007: the University of Amsterdam's Question Answering System. After question analysis, a question is forwarded to two table modules and two retrieval modules, all of which generate candidate answers. These four question processing streams use the two data sources for this task, the AQUAINT newspaper corpus and the blog corpus, as well as fact tables which were generated from these data sources, and the Web. Related candidate answers are combined and ranked by a postprocessing module, which produces the final list of answers to the question.

collection presented a number of challenges that are typical for web corpora, such as encoding issues, mix of languages, boilerplate text (e.g. advertisement), etc. In addition, the size of the blog corpus is much larger than previous corpora used for TREC QA.

To address these challenges we preprocess the blog corpus using a set of Perl scripts. First, we detect the character encoding of each blog post and convert all characters to UTF-8. Second, we detect the language of the blog post using TextCat [3] and remove all non-English blog posts. Third, we attempt to extract the content of the blog post using a set of templates[1]. If template matching fails, we fall back to extracting post titles and removing HTML tags only. Finally, the blog post content is split into sentences using a rule-based sentence splitter[2].

Our content extraction method using templates was based on the observation that a large portion of the blog posts was published using major blog publishing platforms. For example, about 38% of the blog posts were published using the platform Blogger. These platforms usually use templates to generate the blog HTML pages. Consequently, it should be possible to extract the original content by reverse engineering the templates used to generate these pages. We manually created templates for the 7 most common publishing platforms, WordPress, Blogger, Typepad, Moveable Type, moveabletype.org, canalblog, and CommunityServer.

The blog corpus comprises more than 3 million blog posts split up over 3247 TREC format files. Due to the large size of the corpus, we had to run preprocessing on the grid, and some files could not be preprocessed in time. We were able to process 86.5% of the blog posts. 12.6% of all posts were classified as non-English posts and were not included in content extraction. Template-based content extraction was successful for only 19.4% of the blog posts. We assume that small changes in templates between different versions of publishing platforms are responsible for this relatively small number. Automatic learning of templates may be a way towards more robust content extraction. For the remaining 54.4% of the blog posts, only title extraction, HTML tag removal and sentence splitting was performed.

[1]Using Template::Extract, available from http://cpan.org.
[2]We use the Perl module Lingua::EN::Sentence (http://cpan.org).

4. QA AS XML RETRIEVAL

One of our QA streams, *XQuesta* implements the "Question Answering as Semistructured Retrieval" approach. We view various text analysis tools (sentence splitter, part-of-speech and named entity taggers, chunker, parsers) as black boxes producing stand-off XML annotations of the input text data, and using XML querying for data access. For incoming questions, the question analysis module generates one or more structural queries that are used to extract answers from relevant passages identified by a passage retrieval engine. In this year's version of the system, we processed the data collection offline, generating a large repository of XML annotations. Since annotation tools are independent, the results generally cannot be presented in a single XML tree: e.g., for each document, we have sentence-split, POS-tagged, chunked, NE-tagged, and syntactically parsed versions in separate XML DOM structures. To facilitate transparent simultaneous access to such multi-level structures we extended Nux, an open-source Java library for XQuery processing. When accessing an XML document with multiple annotations, we load separate XML files containing layers of stand-off annotation of the same data, and join them in a single new XML object (DOM tree). The new document can be accessed with the standard XQuery facilities. Additionally, we implement an XQuery function `stand-off:wide` that retrieves XML elements overlapping (character-wise) with the current element, irrespective of the annotation layer, based on offsets specified with start and end attributes. For example, the following XQuery extracts person names that occur as parts of syntactic objects of verbs:

```
stand-off:wide(//phrase[@type="VP"]/phrase[@type="NP"])
           /NE[@type="PERSON"]
```

This implementation was inspired by the `select-wide` XPath axis in the XQuery support in the MonetDB [8]. See [2] for more details.

5. REWRITING INTERFACE CODE

The QA system we used in previous years consisted of many components but was mostly developed ad-hoc, i.e., without a consistent system architecture. As a result, the system was difficult to maintain and change. To address this problem we re-implemented large parts of the system following a modular system design. The goal is to develop a self-contained system that is consistent and can be maintained more easily. The main feature of the newly developed system architecture is that it consists of several modules which are cleanly separated by interfaces. This allows us to minimize dependencies between components. A detailed description of the new architecture can be found in [6]

6. LEARNING TO FIND ANSWERS

As described in section 2, our offline information extraction module creates a database of simple relational facts to be used during question answering. The TableLookup QA stream uses a set of manually defined rules to map an analyzed incoming question into a database query. A new stream, *MLTableLookup*, uses supervised machine learning to train a classifier that performs this mapping. In this section we give an overview of our approach. We refer to [7] for further details.

The purpose of the table lookup stream is to map an incoming question Q to an SQL-like query "select AF from T where $sim(QF,Q)$", where AF and QF are fields of table T with QF being the field with information similar to that in the question while field AF contains a candidate answer. In this query formalism, the task of generating the query is a combination of two subtasks: (1) mapping an incoming question Q to a triple $\langle T, QF, AF \rangle$ (a *table-lookup label*) and (2) defining an appropriate similarity function $sim(QF,Q)$, a task for which we use Lucene's vector space model.

The interesting and novel part of the stream is the query formulation part, i.e., training a classifier to predict table lookup labels. For this purpose, features were extracted from the questions from the TREC QA tasks of 2001-2003 with the Quartz question classification module. Pairs of features and answers were retrieved from table rows and these were used as training material for label prediction. A ten-fold cross validation test with this data set reached MRR scores of up to 0.287 which is good given that the tabular data only contains answers to a fraction of the questions.

7. SUBMITTED RUNS

We have submitted three different runs for the main task of the Question Answering track of TREC 2007. In all the runs, list questions are treated as factoid questions by supplying only one answer. For answering OTHER questions we employ our system developed for the WebCLEF 2007 task [4]. For a given topic, the system retrieves relevant passages, splits text in sentences, and estimates importance of sentences for the topic by computing a simple sentence centrality score based on lexical similarity with other sentences in the pool.

The first run (uams07main) was generated by the system described in section 2. In test runs of this system we had noticed two problems. First, although each stream performs internal answer type checks, the system frequently returned answers of the wrong semantic type, like *1792* in response to *Who is Mozart?*. The second problem was related to the final reranking module which was based on web frequency counts of the candidate answers. It displayed erratic behavior and sometimes returned orderings which seemed to be random.

In order to evaluate the effect of these two problems, we created two alternative runs. The first (uams07atch) was generated by applying an extra type check to the answers of the main run, filtering out answers that did not match the expected answer type according to a coarse-grained named entity scheme (classes: person, location, organization, miscellaneous, timex, quantity and other). This run selected the highest ranked answer that passed the filter, or NIL if no such answer was available.

For the second alternative run (uams07nwrr), we used the answers as ranked before running the web ranking module. We did not apply an extra final answer type check here since this would have made it more difficult to find out the exact effect of the reranking module. The additional type check would have had an influence on the result as well.

The results of the three runs can be found in Table 1. Our main interest was the exact factoid accuracy. The main run performed worst (0.072). The run with extra answer type checks (uams07atch, 0.083) achieved approximately 20% relative improvement over the main run. The run without web reranking (uams07nwrr, 0.092) performed best, with a relative improvement of 46.2% over the main run.

The scores for list questions were comparable. Results for OTHER questions were consistently above the median score of all participants. Since we submitted the same answers to these questions in all three runs, we were surprised that there were differences between the scores. According to the evaluation organizers, these are caused by inconsistencies between assessors.

Since the submission deadline, we have performed additional experiments to evaluate different parts of our system. For the evaluations we have only looked at factoid questions of the main track of TREC QA 2007. The unofficial answer patterns for these questions[3] were used to classify answers as correct and incorrect. We were interested in finding answers to the following questions:

1. The QA system performed better with either added answer type check or omitted web reranking. How well would it perform if both modifications were applied at the same time?
2. The system contains six independent QA streams. How well did each of these streams perform on the 2007 factoid questions?
3. The answers of the streams were generated by combining all streams. Can a combination of a subset of these streams achieve a better performance?

The results of the additional evaluation experiments can be found in Table 2. We did not check justification, so the factoid accuracy scores should be interpreted as an approximation of the sum of the exact and unsupported scores of Table 1.

In the first experiment, we performed an extra type check on the answers of the uams07nwrr no web reranking run. According to the factoid answer patterns, this approach achieved an accuracy of 0.168 which constitutes a 43.6% relative improvement over our best submitted run (Table 2). Even with the two modifications, the performances of three of the individual streams (XQUESTA blog and the two table streams) are worse than our lowest-scoring submitted run. The two AQUAINT streams reach a similar performance as our best submitted run while the web ngrams stream performs a lot better. However, it should be mentioned that the last stream does not generate corpus justification for its answers, so using only this stream would lead to all answers being unsupported.

In the final experiment, we selected the top answers from the system that were supported by at least one of a combination of individual QA streams. We used the top three performing individual streams, web ngrams, XQUESTA AQUAINT and AQUAINT ngrams. For answers supported by web ngrams, we additionally required that they were supported by one of the other five QA streams in order to make sure that the answer could be found in the corpus.

[3]http://ilps.science.uva.nl/∼erikt/trec2007/patterns2007.txt

run	factoid accuracy (exact,local,unsup.,inex.)	list F	other F	overall
`uams07main`	0.072 , 0.003 , 0.011 , 0.056	0.032	0.209	0.104
`uams07atch`	0.083 , 0.003 , 0.019 , 0.061	0.032	0.191	0.103
`uams07nwrr`	0.092 , 0.006 , 0.036 , 0.064	0.028	0.198	0.105
median scores	0.131	0.085	0.118	0.108

Table 1: Results for the main QA task: our three submitted runs as well as the median scores of all 51 submitted runs. Our three runs used the same answers for other questions (scoring differences are caused by assessor inconsistencies). Our main system (uams07main) generates more correct factoid answers with an additional answer type check (uams07atch). The best results were achieved when the final web reranking based step was omitted (uams07nwrr).

factoid accuracy exact+unsup.	run description
0.080	uams07main: main run
0.096	uams07atch: with extra type checks
0.117	uams07nwrr: without web reranking
0.168	nwrr + extra type checks
0.028	learned tables (6)
0.045	baseline: always answer NIL
0.048	tables with rules (5)
0.056	XQUESTA blog (4)
0.111	AQUAINT ngrams (3)
0.123	XQUESTA AQUAINT (2)
0.163	web ngrams (1)
0.154	combination: (2) + (3)
0.163	combination: (1) + (3)
0.166	combination: (1) + (2) + (3)
0.181	combination: (1) + (2)

Table 2: Approximate factoid accuracy scores (exact+unsupported) for post-deadline experiments. First, the three submitted runs followed by the best submitted run with additional answer type checks. Then the performances of the six individual streams as well as that of a baseline system which always answers NIL. Finally, an evaluation of four combinations of the best three individual QA streams. The final combination of XQUESTA AQUAINT and web ngrams performs best.

We examined one combination of three streams and three combination of two streams. This approach led to the highest factoid performance score we have measured: 0.181 for the combination of web ngrams and XQUESTA AQUAINT (Table 2).

8. CONCLUSIONS

We described our participation in the main task of the TREC 2007 Question Answering track. This year, we have continued with our approach of QA-as-XML-retrieval setting, where incoming questions are converted to semistructured queries which are applied to a target collection which was automatically annotated with linguistic information at indexing time. Additionally, we added the new blog corpus to the target collection, built a new QA stream for tabular data accessed via learned associations and completely rewrote the architecture code of our QA system.

Our prime goal was to experiment with answer type checking, web re-ranking, and combinations of streams, and to evaluate our new table lookup stream. Our experiments show that answer type checking can substantially improve performance, while web re-ranking can hurt the performance of our system. Substantial improvements also resulted from post-deadline experiments combining different subsets of individual streams. These combinations also outperformed each individual stream. Performance of our new table lookup stream stayed far below the results obtained in cross validation on previous year's data and further analysis will be necessary to fully assess the factors influencing its performance.

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, 612.069.006 640.001.501, 640.002.501, and C.2324.-0114.

References

[1] D. Ahn, S. Fissaha, V. Jijkoun, K. Müller, M. de Rijke, and E. Tjong Kim Sang. Towards a multi-stream question answering-as-xml-retrieval strategy. In *Proceedings of the Fourteenth Text Retrieval Conference (TREC 2005)*. NIST, 2006.

[2] W. Alink, V. Jijkoun, D. Ahn, M. de Rijke, P. Boncz, and A. de Vries. Representing and querying multi-dimensional markup for question answering. In *Proceedings of the 5th Workshop on NLP and XML (NLPXML-2006): Multi-Dimensional Markup in Natural Language Processing*, 2006.

[3] W. Cavnar and J. Trenkle. N-Gram-Based Text Categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994. http://www.let.rug.nl/∼vannoord/TextCat/.

[4] V. Jijkoun and M. de Rijke. The University of Amsterdam at WebCLEF 2007: Using centrality to rank web snippets. In A. Nardi and C. Peters, editors, *Working Notes for the CLEF 2007 Workshop*, 2007. URL: `http://www.clef-campaign.org/2007/working_notes`.

[5] V. Jijkoun, J. van Rantwijk, D. Ahn, E. Tjong Kim Sang, and M. de Rijke. The University of Amsterdam at QA@CLEF 2006. In *Working Notes for the CLEF 2006 Workshop*. Alicante, Spain, 2006.

[6] V. Jijkoun, K. Hofmann, D. Ahn, M. A. Khalid, J. van Rantwijk, M. de Rijke, and E. Tjong Kim Sang. The university of amsterdam at clef@qa 2007. In *Working Notes for the CLEF 2007 Workshop*. Budapest, Hungary, 2007.

[7] M. Khalid, V. Jijkoun, and M. de Rijke. Machine learning for question answering from tabular data. In *FlexDBIST-07 Second International Workshop on Flexible Database and Information Systems Technology*, 2007.

[8] MonetDB, 2007. URL: `http://www.monetdb.nl`.