The University of Amsterdam at CLEF@QA 2007

Valentin Jijkoun Katja Hofmann David Ahn Mahboob Alam Khalid
Joris van Rantwijk Maarten de Rijke Erik Tjong Kim Sang
ISLA, University of Amsterdam
jijkoun,khofmann,ahn,mahboob,rantwijk,mdr,erikt@science.uva.nl

Abstract

We describe a new version of our question answering system, which was applied to the questions of the 2007 CLEF Question Answering Dutch monolingual task. This year, we made three major modifications to the system: (1) we added the contents of Wikipedia to the document collection and the answer tables; (2) we completely rewrote the module interface code in Java; and (3) we included a new table stream which returned answer candidates based on information which was *learned* from question-answer pairs. Unfortunately, the changes did not lead to improved performance. Unsolved technical problems at the time of the deadline have led to missing justifications for a large number of answers in our submission. Our single run obtained an accuracy of only 8% with an additional 12% of unsupported answers (last year, our best run achieved 21%).

1 Introduction

For our earlier participations in the CLEF question answering track (2003–2006), we have developed a parallel question answering architecture in which candidate answers to a question are generated by different competing strategies, *QA streams* [4]. Although our streams use different approaches to answer extraction and generation, they share the mechanism for accessing the collection data: we have converted all of our data resources (text, linguistic annotations, and tables of automatically extacted facts) to fit in an XML database in order to standardize the access [4]. For the 2007 version of the system, we have focused on three tasks:

1. Add to the data resources of the system material derived from the Dutch Wikipedia (previously only derived from Dutch newspaper text).

2. Rewrite the out-of-date code which takes care of the communication between the different modules (previously in Perl) in Java. In the long run we are aiming at a system which is completely written in Java and is easily maintainable.

3. Add a new question answering stream to our parallel architecture: a stream that generates answers from pre-extracted relational information based on *learned* associations between questions and answers; a similar stream in last year's system used manual rules for identifying such associations.

This paper is divided in seven sections. In section 2, we give an overview of the current system architecture. In the next three sections, we describe the changes made to our system for this year: resource adaptation (section 3), code rewriting (4), and the new table stream (5). We describe our submitted runs in section 6 and conclude in section 7.

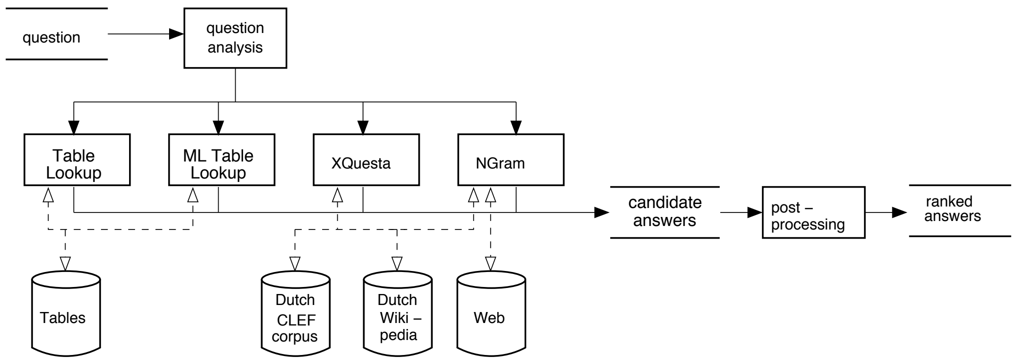

Figure 1: Quartz-2007: the University of Amsterdam's Dutch Question Answering System. After question analysis, a question is forwarded to two table modules and two retrieval modules, all of which generate candidate answers. These four question processing streams use the two data sources for this task, the Dutch CLEF newspaper corpus and Dutch Wikipedia, 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.

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 text corpus that they employ: the CLEF-QA corpus, Dutch Wikipedia, or the Web. Below, we describe these briefly.

The Quartz system (Figure 1) contains four streams that generate answers from the two CLEF data sources, the CLEF newspaper corpus and Wikipedia. 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 ways. The stream uses the analysis of a question to determine how a candidate answer should be looked up in the database using a manually defined mapping from question 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 5 for a more elaborate description). The *Ngrams* stream looks for answers in the corpus by searching for most frequent word ngrams in a list of text passages retrieved from the collection using a standard retrieval engine (Lucene) using a text query generated from the question.

The most advanced of the four streams is XQuesta. For a given question, it automatically generates XPath queries for answer extraction, and executes them on an XML version of the corpus which contains both the corpus text and additional annotations. The annotations include information about part-of-speech, syntactic chunks, named entities, temporal expressions, and dependency parses (from the Alpino parser [6]). For each question, XQuesta only examines text passages relevant to the question (as identified by Lucene).

There is one stream which employs textual data outside the CLEF document collection defined for the task: the *Ngrams* 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 answers candidates found by this approach are not backed up by documents from the CLEF 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.

3 Wikipedia as a QA Resource

Our system uses Dutch Wikipedia in the same way as the Dutch newspaper corpus. We used an XML dump of Wikipedia[1] that provides basic structural markup and additionally annotated it with sentence boundaries, part-of-speech tags, named entities and temporal expressions. Wikipedia was consulted by the XQuesta and NGram streams and was also used for offline information extraction.

4 Rewriting Interface Code in Java

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.

Figure 2 gives an overview of the main components of our QA system. The core modules are *document_collection*, *text_analysis*, *data*, *question_answering*, and *answer_selection*. The package *apps* contains several small programs that combine functionality of the core modules into complete applications, such as the CLEF batch system, an interactive command-line program, and an online demo of our QA system.[2]

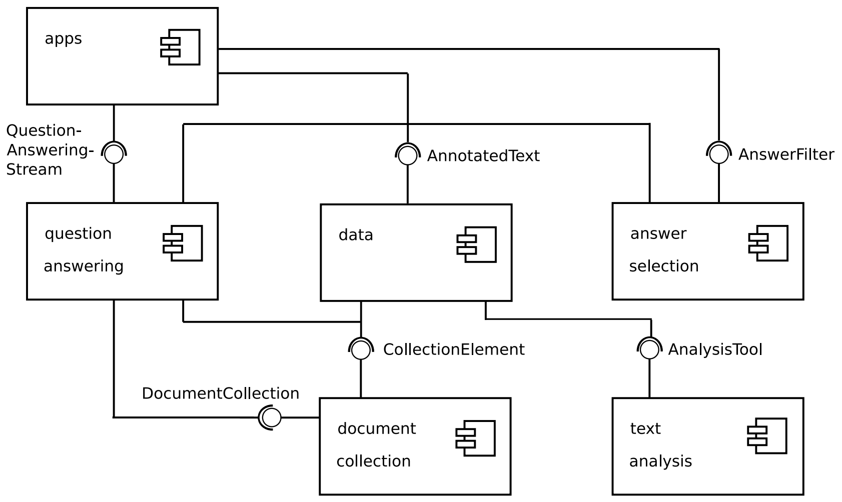

Figure 2: UML Component diagram showing modules and module dependencies. Boxes represent individual modules, circles represent interfaces, and half-circles indicate that a module depends on an interface.

Central to our system is the *data* module which abstracts all textual data within the system as *AnnotatedText*. *AnnotatedText* objects maintain an XML representation of the data and allow access through both Java methods and XQuery. XML annotations can be added as necessary. At

[1]URL: `http://ilps.science.uva.nl/WikiXML`

[2]The Dutch version of our QA demo can be accessed at `http://cs-ilps.science.uva.nl:20500/`. Uses Table Lookup stream only.

each processing step the data objects can be serialized to their XML representation for logging or data exchange with external programs.

The *question_answering* module contains our QA streams. Each stream implements the interface *QuestionAnsweringStream* which allows applications to run streams in a unified way. This allows us to add new QA streams to the system on the fly, without changing any of the existing components.

To make question answering streams independent of the specifics of different document collections — SQL tables, the CLEF newspaper corpus, Wikipedia, and the web — the *document_collection* module provides access via the *DocumentCollection* and *CollectionElement* interfaces. *DocumentCollection* provides methods for retrieving elements from a collection. Implementations for different IR engines and web search engines were developed. To create answer candidates from CollectionElements, the elements are converted into AnnotatedText.

Text analysis tools, such as part-of-speach tagger, named-entity tagger, and question classifier, are part of the *text_analysis* module and implement the *AnalysisTool* interface. Analysis tools are run on demand by a mechanism provided by the *data* module. Consumers of data objects, such as QA streams, specify which tools they require. The tools produce XML annotations which are added to the data objects and can be queried, for example using XQuery. We attempt to use standard XML annotation formats whenever possible.

The *answer_selection* module contains filters for post-processing lists of answer candidates. Each post-processing tool implements the interface *AnswerFilter* so that applications using these tools are independent of implementation details. Sequences of post-processing tools can be assembled by higher-level applications as necessary. The tools can be applied either per-stream, or to the combined output of the system as a whole.

5 Machine Learning for QA from Tabular Data

As described in section 2, our offline information extraction module creates a database of simple relational facts to be used during question answering. A *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 [5] for further details.

Essentially, the purpose of the table lookup stream is to map an incoming question to an SQL-like query "select AF from T where $sim(QF, Q)$", where T is the table that contains the answer in field AF and its other field QF has a high similarity with the input question Q. Executing such a query for a given question results in a list of answer candidates—the output of the *MLTableLookup* stream.

In the query formalism described above, the task of generating the query can be seen as the task of mapping an incoming question Q to a triple $\langle T, QF, AF\rangle$ (a *table-lookup label*) and defining an appropriate similarity function $sim(QF, Q)$.

The database of facts extracted from the CLEF QA collection consists of of 16 tables containing 1.4M rows in total. For example, the *Definitions(name, definition)* table contains the definition of *Soekarno* as *president of Indonesia*, the table *Birthdates(name,birthdate)* contains the information that *Aleksandr Poesjkin* was born in *1799*. Then, for a question such as *Wie was de eerste Europese commissaris uit Finland? (Who was the first European Commissioner from Finland?)* the classifier may assign the table-lookup label $\langle T : Definitions, QF : Definition, AF : name\rangle$. In this case, the question would be mapped to the SQL like query "select $name$ from $Definitions$ where $sim(definition, \{eerste, European, commissaris, Finland\})$".

For an incoming question, we first extract features and apply a statistical classifier that assign a *table-lookup* label, i.e., a triple $\langle T, QF, AF\rangle$. We then use a retrieval engine to locate values of field QF in table T which are most similar to the text of the question Q (according to a retrieval function $sim(\cdot, \cdot)$), and return values of corresponding AF fields. Figure 3 shows the architecture of our system.

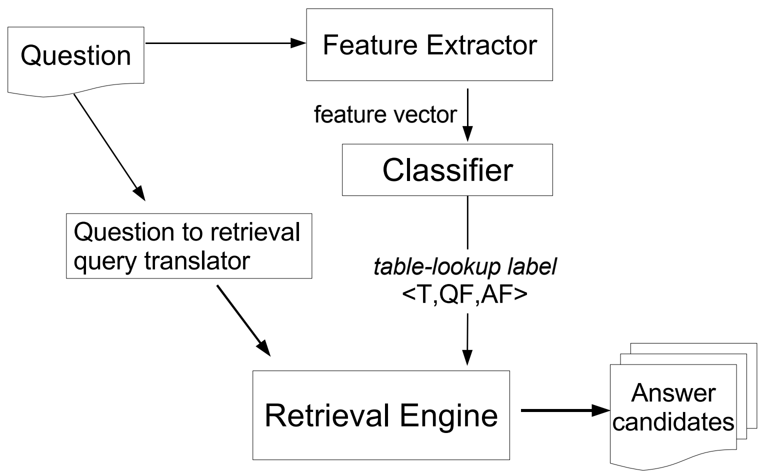

Figure 3: Architecture of the MLTableLookup QA stream.

Our architecture depends on two modules: the classifier that predicts table lookup labels and the retrieval model $sim(\cdot, \cdot)$ along with the text representation and the retrieval query formulation. For the later task we selected Lucene's vector space model as retrieval model, and used a combination of two types of text representation, exact and stemmed forms of the question words, to formulate a retrieval query, i.e., to translate an incoming to question to a retrieval query.

The interesting and novel part of the new QA system is the second stage of our query formulation, i.e., training a classifier to predict table lookup labels. This stage, in turn, can be split in two parts: generating training data and actually training the classifier. We generate the training data using the selected retrieval model. We index the values of all fields of all rows in our database as separate documents. For each question q, we translate the question into the retrieval query and use the selected retrieval model to generate a ranked list of field values from our database. We select the document, table name T and the field name QF, such that it occurs first time in the ranked list and the value of some other field AF contains the answer of the question. In other words we find T, QF and AF such that the query "select AF from T where $sim(QF, Q)$" returns a correct answer to question q at the top rank. We use the label $\langle T, QF, AF\rangle$ as a correct class for question q. For example, we translate the question *In welk land in Afrika is een nieuwe grondwet aangenomen?* (whose answer is *Zuid-Afrika*) into a retrieval query that is composed of the question words and words retrieved from the process of filtering out stopwords and stem the remaining the question words. Then we run the query against the retrieval engine's index; for this particular example our system finds the triplet $\langle T : Locations, QF : location_b, AF : location_a\rangle$ as the optimal table-lookup label for this question.

Next, in order to generate training data, we represent each question as a set of features. We use the existing module of [2] to construct the set of features. Finally we train a memory-based classifier TiMBL [1] and use a parameter optimization tool to find the best setting for Timbl; see Figure 4 for an overview.

We used a set of question/answer pairs from the CLEF-QA tasks 2003–2006 and a knowledge base with tables extracted from the CLEF-QA corpus using the information extraction tools of QUARTZ system. We split our training corpus of 644 questions with answers into 10 sets and run a 10-fold cross-validation. The performance of the system is measured using the Mean Reciprocal Rank (MRR, the inverse of the rank of the first correct answer, averaged over all questions) and accuracy at n ($a@n$, the number of question answered at rank $\leq n$). Table 1 shows the evaluation results averaged over the 10 folds.

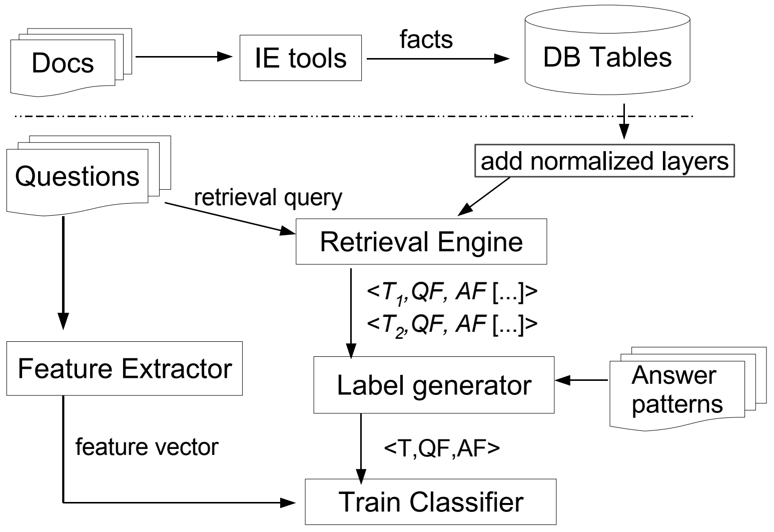

Figure 4: Learning table lookup labels

a@1	a@5	a@10	MRR
13.1%	21.4%	24.1%	0.593

Table 1: Evaluation of the ML Table-lookup QA stream applied to the CLEF 2003–2006 question answer pairs.

6 Runs

We have submitted a single Dutch monolingual run: uams071qrtz. The associated evaluation results can be found in Table 2. In this run, we have treated list questions as factoid questions: always returning the top answer. The planned updates of the system proved to be more time consuming than was expected. The system was barely finished at the time of the deadline. Because of this there was no time for an elaborate test or for compiling alternative runs. The performance of the system has suffered from this: only about 8% of the questions were answered correctly. The previous version achieved 21% correct on the CLEF-2006 questions.

The prime cause of the performance drop can be found in the submitted answer file. No less than 81 (41%) of the 200 answers did not contain the required answer snippet. This problem caused all but 4 of the unsupported assessments. 22 of these 81 answers mentioned a document id for the missing snippet but the other 59 lacked the id as well. The problem was caused by a mismatch between the new java code and the justification module which caused all justifications associated with answers from the two table streams to be lost.

When examining the answers for the factoid and definition questions, we noticed that a major problem is a mismatch between the expected answer type and the type of the answer. Here are a few examples:

0003. How often was John Lennon hit? Answer: Yoko Ono
0136. What is an antipope? Answer: Anacletus II
0160. Who is Gert Muller? Answer: 1947

As many as 61 of the 161 incorrectly answered displayed such a type mismatch. The question classification part of the system (accuracy: 80%) generates an expected type for each answer but it is not used in the postprosessing phase. Indeed, the addition of a type-based filter at the end of the processing phase is one of the most urgent tasks for future work.

Question type	Total	Right	Unsupported	Inexact	Wrong	% Correct
factoid	156	14	17	0	125	9%
definition	28	1	5	0	22	4%
factoid+definition	184	15	22	0	147	8%
list	16	0	1	1	14	0%
temporarily restricted	41	2	3	0	36	5%
unrestricted	159	13	20	1	125	8%
all	200	15	23	1	161	8%

Table 2: Assessment counts for the 200 top answers in the Amsterdam run submitted for Dutch monolingual Question Answering (NLNL) in CLEF-2007. About 8% of the questions were answered correctly. Another 12% were correct but insufficiently supported. The run did not contain NIL answers.

7 Conclusion

We have described the fifth iteration of our system for the CLEF Question Answering Dutch mono-lingual track (2007). While keeping the general multi-stream architecture, we re-designed and re-implemented the system in Java. This was an important update, which however did not lead to improved performance, mainly due to many technical problems that were not solved by the time of the deadline. In particular, these problems led to originating snippet being lost for many of the answer candidates extracted from the collection, leading to a large number of answers in our submission. Our single run obtained an accuracy of only 8% with an additional 12% of unsupported answers (last year, our best run achieved 21%).

Addressing these issues, performing a more systematic error analysis and answer extraction step in XQuesta stream and learning step in MLTableLookup are the most important items for future work.

Acknowledgments

This research was supported by various grants from the Netherlands Organisation for Scientific Research (NWO). Valentin Jijkoun was supported under project numbers 220.80.001, 600.065.120 and 612.000.106. Joris van Rantwijk and David Ahn were supported under project number 612.066.302. Erik Tjong Kim Sang was supported under project number 264.70.050. Maarten de Rijke was supported by NWO under project numbers 017.001.190, 220.80.001, 264.70.050, 354.20.005, 600.065.120, 612.13.001, 612.000.106, 612.066.302, 612.069.006, 640.001.501, and 640.-002.501. Mahboob Alam Khalid and Katja Hofmann were supported by NWO under project number 612.066.512.

References

[1] Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. *TiMBL: Tilburg Memory Based Learner, version 5.1, Reference Guide*. University of Tilburg, ILK Technical Report ILK-0402., 2004. http://ilk.uvt.nl/.

[2] V. Jijkoun, G. Mishne, and M. de Rijke. Building infrastructure for Dutch question answering. In *Proceedings DIR-2003*, 2003.

[3] Valentin Jijkoun and Maarten de Rijke. Retrieving answers from frequently asked questions pages on the web. In *Proceedings of the Fourteenth ACM conference on Information and knowledge management (CIKM 2005)*. ACM Press, 2005.

[4] Valentin Jijkoun, Joris van Rantwijk, David Ahn, Erik Tjong Kim Sang, and Maarten de Rijke. The University of Amsterdam at QA@CLEF 2006. In *Working Notes for the CLEF 2006 Workshop*. Alicante, Spain, 2006.

[5] M.A. 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.

[6] Gertjan van Noord. At last parsing is now operational. In *Proceedings of TALN 2006*. Leuven, Belgium, 2006.