Towards an Offline XML-based Strategy for Answering Questions

David Ahn, Valentin Jijkoun, Karin Müller
Maarten de Rijke, and Erik Tjong Kim Sang

ISLA, University of Amsterdam, Kruislaan 403
1098 SJ Amsterdam, The Netherlands
`{ahn, jijkoun, kmueller, mdr, erikt}@science.uva.nl`

Abstract. The University of Amsterdam participated in the Question Answering (QA) Track of CLEF 2005 with two runs. In comparison with previous years, our focus this year was adding to our multi-stream architecture a new stream that uses offline XML annotation of the corpus. We describe the new work on our QA system, present the results of our official runs, and note areas for improvement based on an error analysis.

1 Introduction

For our participation in question answering (QA) tracks at past editions of both CLEF and TREC, we have developed a *multi-stream* QA architecture which incorporates several different approaches to identifying candidate answers, complemented with filtering and ranking mechanisms to choose the best answer [1, 2, 3, 4]. For the 2005 edition of the QA@CLEF track, we devoted some effort to improving this architecture, in particular the table stream (see §2.2). Also, to accommodate the new temporally restricted questions, a dedicated module was developed (§2.3). Most of our efforts, however, were aimed at implementing XQuesta, a pure QA-as-XML-retrieval stream, in which the target collection is automatically annotated with linguistic information at indexing time, incoming questions are converted to semistructured queries, and evaluation of these queries yields a ranked list of candidate answers.

While our system provides *wrong* answers for less than 40% of the test questions, we identified obvious areas for improvement. First, we should work on definition extraction so that both questions asking for definitions and questions requiring resolving definitions can be better answered. Second, we should examine inheritance of document links in the answer tiling process to help the associated module avoid unsupported answers. Most importantly, we should improve our answer filtering module to ensure that the semantic class of the generated answer corresponds with the class required by the question.

The paper is organized as follows. In §2, we describe the architecture of our QA system, including improvements and additions for QA@CLEF 2005. In §3, we describe the new XQuesta stream. In §4, we detail our official runs. In §4, we discuss the results we obtained and give a preliminary analysis of the performance of different components of the system. We conclude in §5.

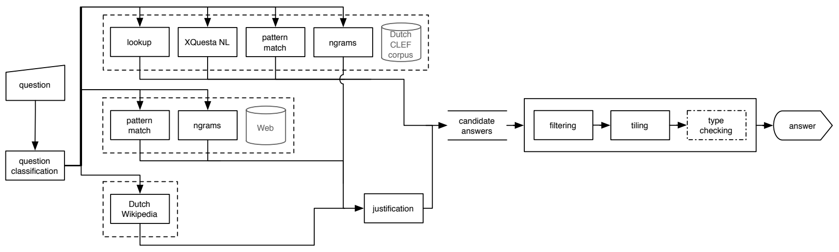

Fig. 1: Quartz-2005: the University of Amsterdam's Dutch Question Answering System.

2 System overview

Essentially, our system architecture implements multiple copies of a standard QA architecture. Each copy (or *stream*) is a complete standalone QA system that produces ranked answers, though 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. For a reasonably detailed discussion of our QA system architecture we refer to [3]. A diagram of the system architecture is given in Figure 1.

The first stage of processing, *question processing*, is common to all the streams. Each of the 200 questions is tagged, parsed, and assigned a question class based on our question classification module. Finally, the expected answer type is determined. See §3.2 for more details about question processing for the XQuesta stream, in particular.

There are seven streams in our system this year, four of which use the CLEF corpus to answer questions and three of which use external sources of information. Four streams are unchanged from our system for last year's evaluation [3]: the *Pattern Match* and *Ngrams* streams for each of the CLEF corpus and the web. The focus of our efforts this year resulted in our XQuesta stream, which is described in §3. We also added a stream that consults Wikipedia (§2.1) and expanded the table stream (§2.2).

The methods we employ to merge, filter, and choose among the answer candidates generated by the seven streams, as well as to *justify* answers (i.e., find supporting documents in the Dutch CLEF corpus for answers obtained outside the corpus), also remain unchanged from our system for last year's evaluation, except for the temporally restricted questions new to this year's evaluation. For these questions, we re-rank candidate answers using temporal information; see §2.3 for more details.

2.1 Wikipedia stream

Like our streams that consult the web rather than the Dutch CLEF corpus, this stream also uses an external corpus—the Dutch Wikipedia (http://nl.wikipedia.org), an open-content encyclopedia in Dutch. However, since this corpus is much cleaner than newspaper text, the stream operates in a different manner. First, the *focus* of the question—usually the main named entity—is identified. Then, this entity's encyclopedia entry is looked up; since Wikipedia is standardized to a large extent, this information has a template-like nature. Finally, using knowledge about the templates used in Wikipedia, information such as Date-of-death and First-name can easily be extracted.

2.2 Improvements to the table stream

To the tables used in 2004, we have added a table which contains definitions extracted offline with two rules: one extracts definitions from appositions, and the other creates definitions by combining proper nouns with preceding common nouns. This table is used in parallel with the existing roles table, which contains definitions only for people. The new table contains more than three times as many entries (611,077) as the existing one.

Unlike earlier versions of the table module, all tables are now stored in SQL format and made available in a MySQL database. The type of an incoming question is converted to sets of tuples containing three elements: table, source field, and target field. The table code searches in the source field of the specified table for a pattern and, when a match is found, keeps the contents of the corresponding target field as a candidate answer. Ideally, the search pattern would be computed by the question analysis module but currently we use separate code for this task. The table code also uses backoff strategies (case insensitive vs. case sensitive, exact vs. inexact match) in case a search returns no matches.

The table fields only contain noun phrases that are present in the text. This means that they can be used for answering questions such as *Who is the President of Serbia?* because phrases such as *President of Serbia* can usually be found in the text. However, in general, this stream cannot be used for answering questions such as *Who was the President of Serbia in 1999?* because the modifier *in 1999* often does not follow the profession.

2.3 Temporal restrictions

Twenty-six questions in this year's QA track are tagged as *temporally restricted*. Such questions ask for information relevant to a particular time; the time in question may be given explicitly by a temporal expression (or *timex*), as in:

(1) *Q0094: Welke voetballer ontving "De Gouden Bal" in 1995?*
Which footballer won the European Footballer of the Year award in 1995?

or it may be given implicitly, with respect to another event, as in:

(2) *Q0008: Wie speelde de rol van Superman voordat hij verlamd werd?*
Who played the role of Superman before he was paralyzed?

Our system takes advantage of these temporal restrictions to re-rank candidate answers for these questions. Because there is already a module to annotate timexes (see §3.1), we limit ourselves to temporal restrictions signalled by timexes. Handling event-based restrictions would require identifying (and possibly temporally locating) events, which is a much more difficult problem.

For each temporally restricted question, the temporal re-ranker tries to identify an explicit temporal restriction by looking for temporal prepositions (e.g., *in*, *op*, *tijdens*, *voor*, *na*) and timexes in the question. If it succeeds, it proceeds with re-ranking the candidate answers.

For each candidate answer, timexes occurring in sentences containing the answer and question focus (if there is one) are extracted from the justification document, along with the document timestamp. The re-ranker checks whether these timexes are compatible with the restriction. For each compatible timex, the score for the candidate answer is boosted; for each incompatible timex, the score is lowered. The logic involved in checking compatibility of a timex with a temporal restriction is relatively straightforward; the only complications come in handling times of differing granularities.

3 XQuesta

The XQuesta stream implements a QA-as-XML-retrieval approach [5, 6]. The target collection is automatically annotated with linguistic information offline. Then, incoming questions are converted to semistructured queries, and evaluation of these queries yields a ranked list of candidate answers. We describe the three stages in detail.

3.1 Offline annotation

We automatically processed the Dutch QA collection, identifying sentences and annotating them syntactically and semantically. We used the TnT tagger [7] to tag the collection for parts of speech and syntactic chunks, with the CGN corpus [8] as training data. The same tagger, trained on CoNLL-2002 data [9] was used to identify named entities, and a hand-coded rule-based system, to identify temporal expressions.

In total, we use four annotation layers. The first layer provides information about part-of-speech tags:

(3) `<LID>`*de*`</LID>` `<ADJ>`*machtige*`</ADJ>` `<N>`*burgemeester*`</N>` `<VZ>`*van*`</VZ>`
the powerful mayor of ...

Example (4) shows the second annotation layer: non-recursive syntactic chunks—noun phrases (NP), verb phrases (VP) and prepositional phrases (PP).

(4) `<NP>`*de machtige burgemeester*`</NP>` `<PP>`*van*`</PP>` `<NP>`*Moskou*`</NP>` *,*
`<NP>`*Joeri Loezjkov*`</NP>` *,* `<VP>`*veroordeelt*`</VP>` `<NP>`*dat*`</NP>`

Example (5) shows the third annotation layer: named entities—persons (PER), organizations (ORG), locations (LOC), and miscellaneous entities (MISC).

(5) *de machtige burgemeester van* `<NE type="LOC">`*Moskou*`</NE>` *,*
`<NE type="PER">`*Joeri Loezjkov*`</NE>` *, veroordeelt dat*

The next two examples show annotation of temporal expressions, normalized to ISO 8601 format.

(6) *Luc Jouret werd in* `<TIMEX val="1947">`*1947*`</TIMEX>` *geboren.*
Luc Jouret was born in 1947.

(7) *Ekeus verliet Irak* `<TIMEX val="1994-10-06">`*donderdagmorgen*`</TIMEX>`
Ekeus left Iraq Thursday morning

Normalization is more complicated in Example (7); in order to determine that *donderdagmorgen* refers to 1994-10-06, the system uses the document timestamp (in this case, 1994-10-08) and some simple heuristics to compute the reference.

The four annotation layers of the collection are stored in separate XML files to simplify maintenance. Whenever the XQuesta stream requests a document from the collection, all annotations are automatically merged into a single XML document providing full simultaneous access to all annotated information.

3.2 Question analysis

The current question analysis module consists of two parts. The first part determines possible question classes, such as DATE_BIRTH for the question shown in Example (8).

(8) *Q0014: Wanneer is Luc Jouret geboren?*
When was Luc Jouret born?

We use 31 different question types, some of which belong to a more general class: for example, DATE_BIRTH and DATE_DEATH are subtypes of the class DATE. The assignment of the classes is based on manually compiled patterns.

The second part of our question analysis module is new. Depending on the predicted question class, an expected answer type is assigned. The latter describes syntactic, lexical or surface requirements to be met by the possible answers. The restrictions are formulated as XPath queries, which are used to extract specific information from our preprocessed documents. E.g., the XPath queries corresponding to question types PERSON and DATE are `NE[@type="PER"]` and `TIMEX[@val=~/^\d/]`, respectively. Table 1 displays the manually developed rules for mapping the question classes to the expected answer types.

3.3 Extracting and ranking answers

As described in §3.2, incoming questions are mapped to retrieval queries (the question text) and XPath queries corresponding to types of expected answers.

Retrieval queries are used to locate relevant passages in the collection. For retrieval, we use nonoverlapping passages of at least 400 characters starting and ending at paragraph boundaries. Then, the question's XPath queries are evaluated on the top 20 retrieved passages, giving lists of XML elements corresponding to candidate answers. For example, for the question in Example (8) above, with the generated XPath query "`TIMEX[@val=~/^\d/]`", the value "*1947*" is extracted from the annotated text in Example (6).

Question class	Restrictions on the type of answer
ABBREVIATION	word in capital letters
AGE	numeric value, possible word: jarige
CAUSE_REASON	sentence
CITY_CAPITAL	LOC
COLOR	adjective
DATE_DEATH, DATE_BIRTH, DATE	TIMEX, digital number
DEFINITION_PERSON	sentence
DEFINITION	noun phrase or sentence
DISTANCE	numeric value
DISTINCTION	noun phrase or a sentence
EXPANSION	MISC or ORG, noun phrase
HEIGHT	numeric value
LANGUAGE	MISC
LENGTH	numeric value
LOCATION	LOC
MANNER	sentence
MONETARY_UNIT	MISC
NAME	named entity
NUMBER_PEOPLE	numeric value, noun phrase
NUMBER	numeric value
ORGANIZATION	ORG
PERSON	PER
SCORE, SIZE, SPEED, SUM_OF_MONEY	numeric value
SYNONYM_NAME	PER
TEMPERATURE, TIME_PERIOD	numeric value

Table 1: Overview of the mapping rules from question classes to answer types

The score of each candidate is calculated as the sum of retrieval scores of all passages containing the candidate. Furthermore, the scores are normalized using web hit counts, producing the final ranked list of XQuesta's answer candidates.

4 Results and analysis

We submitted two Dutch monolingual runs. The run uams051nlnl used the full system with all streams described above and final answer selection, while uams052nlnl, on top of this, used an additional stream: the XQuesta stream with paraphrased questions. We generated paraphrases simply, by double-translating questions (from Dutch to English and then back to Dutch) using Systran, an automatic MT system. Question paraphrases were only used for query formulation at the retrieval step; question analysis (identification of question types, expected answer types and corresponding XPath queries) was performed on the original questions. Our idea was to see whether paraphrasing retrieval queries would help to find different relevant passages and lead to more correctly answered questions.

The two runs proved to be quite similar. Different answers were only generated for 13 of the 200 questions. The results of the assessment were even more similar. Both runs had 88 correct answers and 5 unsupported answers. Run uams051nlnl had one less inexact answer than uams052nlnl (28 vs. 29) and one more wrong answer (79 vs. 78). We were surprised about the large number of inexact answers. When we examined the inexact answers of the first run, we found that a disproportional number of these were generated for definition questions: 85% (only 30% of the questions ask for definitions). Almost half of the errors (13 out of 28) were caused by the same problem: determining where a noun phrase starts, for example, *leader of the extreme right group* as an answer to *What is Eyal?* where *extreme right group* would have been correct. This extraction problem also affected the answers for questions that provided a definition and asked for a name. We expect that this problem can be solved by a check of the semantic class of the question focus word and head noun of the answer, both in the answer extraction process and in answer postprocessing. Such a check would also have prevented seven of the 15 other incorrect answers of this group.

When we examined the assessments of the answers to the three different question types, we noticed that the proportion of correct answers was the same for definition questions (45%) and factoid questions (47%) but that temporally restricted questions seemed to cause problems (27% correct). Of the 18 incorrect answers in the latter group, four involved an answer which would have been correct in another time period (questions Q0078, Q0084, Q0092 and Q0195). If these questions had been answered correctly, the score for this category would have an acceptable 46% (including the incorrectly assessed answer for Q0149).

The temporal re-ranking module described in §2 did make a small positive contribution. For two temporally restricted questions, the highest ranking candidate answer before the temporal re-ranking module was applied was incorrect, but the application of the temporal re-ranking module boosted the correct answer to the top position. Additionally, the temporal re-ranking module never demoted a correct answer from the top position.

The answers to the temporally restricted questions are indicative of the overall problems of the system. Of the other 14 incorrect answers, only five were of the expected answer category while nine were of a different category. In the 62 answers to factoid and definition questions that were judged to be wrong, the majority (58%) had an incorrect answer class. An extra answer postprocessing filter that compares the semantic category of the answer and the one expected by the question would prevent such mismatches.

Our system produced five answers which were judged to be unsupported. One of these was wrong, one was right, and a third was probably combined from different answers, with a link to a document containing only a part of the answer being kept. The remaining two errors were probably also caused by a document link which should not have been kept but the reason for this is unknown.

This error analysis suggests three possible improvements. First, we should work on definition extraction so that questions asking for definitions and questions requiring the resolution of definitions can be better answered. Second, we should examine inheritance of document links in the answer tiling process to make sure that the associated module avoids unsupported answers. Most importantly, we should improve answer filtering to make sure that the semantic class of the generated answer corresponds with the class required by the question.

5 Conclusion

Most of our efforts for the 2005 edition of the QA@CLEF track were aimed at implementing XQuesta, a "pure" QA-as-XML-retrieval stream, as part of our multi-stream question answering architecture. For XQuesta, the target collection is automatically annotated with linguistic information at indexing time, incoming questions are converted to semistructured queries, and evaluation of these queries gives a ranked list of candidate answers. The overall system provides *wrong* answers for less than 40% of the questions. Our ongoing work is aimed at addressing the main sources of error: definition extraction, inheritance of document links in answer tiling, and semantically informed answer filtering.

Acknowledgments

This research was supported by various grants from the Netherlands Organization for Scientific Research (NWO). Valentin Jijkoun, Karin Müller, and Maarten de Rijke were supported under project number 220-80-001. Erik Tjong Kim Sang and Maarten de Rijke were supported under project number 264-70-050. David Ahn and Maarten de Rijke were supported under project number 612.066.302. In addition, Maarten de Rijke was also supported by grants from NWO, under project numbers 017.001.190, 365-20-005, 612-13-001, 612.000.106, 612.000.207, 612.069.006, and 640.001.501.

6 References

[1] Jijkoun, V., Mishne, G., de Rijke, M.: How frogs built the Berlin Wall. In: Proceedings CLEF 2003. LNCS, Springer (2004)

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

[3] Ahn, D., Jijkoun, V., Müller, K., de Rijke, M., Schlobach, S., Mishne, G.: Making stone soup: Evaluating a recall-oriented multi-stream question answering stream for dutch. In Peters, C., Clough, P., Jones, G., Gonzalo, J., Kluck, M., Magnini, B., eds.: Multilingual Information Access for Text, Speech and Images: Results of the Fifth CLEF Evaluation Campaign. LNCS 3491, Springer Verlag (2005)

[4] Ahn, D., Jijkoun, V., Mishne, G., Müller, K., de Rijke, M., Schlobach, S.: Using wikipedia at the trec qa track. In Voorhees, E., Buckland, L., eds.: The Thirteenth Text Retrieval Conference (TREC 2004), Gaithersburg, Maryland (2005)

[5] Litkowksi, K.: Use of metadata for question answering and novelty tasks. In: Proceedings of the Twelfth Text REtrieval Conference (TREC 2003). (2004)

[6] Ogilvie, P.: Retrieval using structure for question answering. In Mihajlovic, V., Hiemstra, D., eds.: Proceedings of the First Twente Data Management Workshop (TDM'04). (2004) 15–23

[7] Brants, T.: TnT – A Statistical Part-Of-Speech tagger. Saarland University (2000)

[8] Schuurman, I., Schouppe, M., Hoekstra, H., van der Wouden, T.: CGN, an Annotated Corpus of Spoken Dutch. In: Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03), Budapest, Hungary (2003)

[9] Tjong Kim Sang, E.F.: Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In: Proceedings of CoNLL-2002, Taipei, Taiwan (2002) 155–158