The University of Amsterdam at CLEF@QA 2006

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

Abstract

We describe the system that generated our submission for the 2006 CLEF Question Answering Dutch monolingual task. Our system for this year's task features entirely new question classification, data storage and access, and answer processing components.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; H.3.7 Digital Libraries; H.2.3 [**Database Managment**]: Languages—*Query Languages*

General Terms

Measurement, Performance, Experimentation

Keywords

Question answering, Questions beyond factoids

1 Introduction

For our earlier participation in the CLEF question answering track (2003–2005), we have developed a question answering architecture in which answers are generated by different competing strategies. For the 2005 edition of CLEF-QA, we focused on converting our text resources to XML in order to make possible a QA-as-XML-retrieval strategy. For 2006, we have converted *all* of our data resources (text, annotations, and tables) to fit in an XML database in order to further standardize access. Additionally, we have devoted attention to improving known weak parts of our system: question classification, type checking, answer clustering, and score estimation.

This paper is divided in nine sections. In section 2, we give an overview of the current system architecture. In the following four sections, we describe the changes we have made to our system for this year: question classification (section 3), storing data with multi-dimensional markup (section 4), and probabilistic answer processing (sections 5 and 6). We present our submitted runs in section 7 and evaluate them in section 8. We conclude in section 9.

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

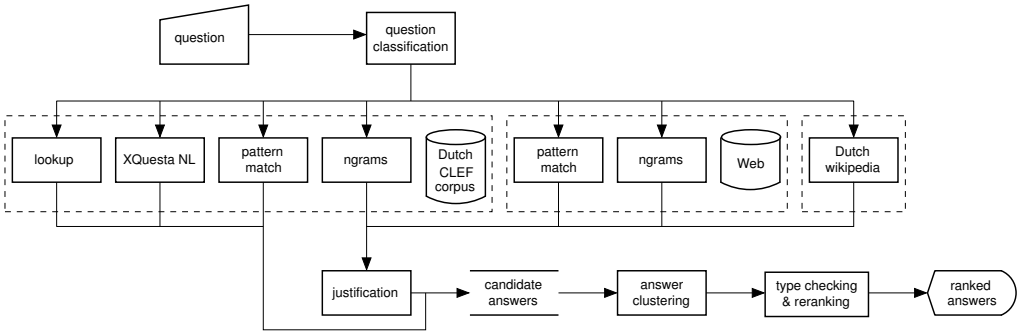


Figure 1: Quartz-2006: the University of Amsterdam's Dutch Question Answering System.

on the text corpus that they employ: the CLEF-QA corpus, Dutch Wikipedia, or the Web. We describe these briefly here. The common question and answer processing parts will be outlined in more detail in the next sections about the architecture updates.

The Quartz system (Figure 1) contains four streams that generate answers from the CLEF-QA 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 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 *Ngrams* stream looks for answers in the corpus by searching for word ngrams using a standard retrieval engine (Lucene). The *Pattern Match* stream is quite similar except that it allows searching for regular expressions rather than just sequences of words.

The most advanced of the four CLEF-QA corpus streams is XQuesta. It performs XPath queries against an XML version of the CLEF-QA 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 [7]). XQuesta retrieves passages of at least 400 characters, starting and ending at paragraph boundaries.

For 2006, we have adapted several parts of our question answering system. We changed the question classification process, the data storage method, and the answer processing module. These topics are discussed in the following sections.

3 Question Classification

An important problem identified in the error analysis of our previous CLEF-QA results was a mismatch between the type of the answer and the required type by the question. A thorough evaluation of the type assignment and type checking parts of the system was required. Over the past three years, our question analysis module—consisting of regular expressions embedded in program code—had grown to a point that it was difficult to maintain. Therefore, we have re-implemented the module from scratch, aiming at classification by machine learning.

We started by collecting training data for the classifier. The 600 questions from the three previous CLEF-QA tracks were an obvious place to start, but restricting ourselves to these questions was not an option, since our goal is to develop a general Dutch QA system. Therefore, we added additional questions: 1000 questions from a Dutch trivia game and 279 questions from other sources—frequently, questions provided by users from our online demo. All the questions have been classified by a single annotator.

Choosing a proper set of question classes was a non-trivial problem. We needed question types not only for extracting candidate answers from text passages but also for selecting columns from tables. Sometimes, a coarse-grained type such as `person` would be sufficient, but for answering *which*-questions, a fine-grained type, such as `American President`, might be better. We therefore

decided on using three different types of question classes: a table type that linked the question to an available table column (17 classes), a coarse-grained type that linked the question to the types recognized by our named-entity recognizer (7 classes), and a fine-grained type that linked the question to WordNet synsets (166 classes).

For the machine learner, a question is represented as a collection of features. We chose ten different features:

1. the question word (e.g., *what* for English or *wat* for Dutch)
2. the main verb of the question
3. the first noun following the question word
4. a normalized version of the noun
5. the top hypernym of the noun according to our type hierarchy
6. the type of the named entity before the noun
7. the type of the named entity after the noun
8. a flag identifying the presence of capital letters
9. a flag identifying a verb-initial question
10. a flag indicating a factoid or a list question

In order to determine the top hypernym of the main noun of the question we use EuroWordNet [8]. We follow the hypernym path until we reach a word that a member of a predefined list which contains entities like *persoon* (*person*) or *organisatie* (*organisation*). The factoid/list flag value is also determined by the number value of the main noun. A plural noun indicates a list question and a singular one, a factoid question. Questions without a noun will be classified by the number of the verb.

A question such as *Who is the new pope?* can be transformed to the features: *who*, *are*, *pope*, *pope*, *pope*, *none*, *none*, 0, 0 and *factoid*. We have not explored other features because with this small set of features we were already able to obtain a satisfactory score for coarse-grained classification (about 90% for the questions from previous CLEF-QA tracks).

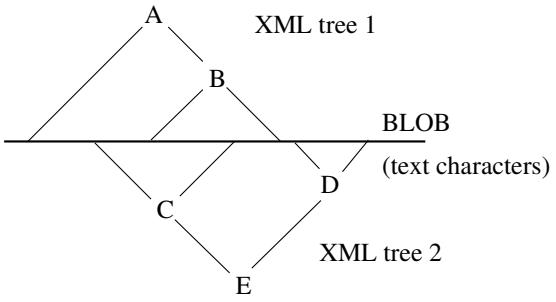
We chose the memory-based learner Timbl [4] for building the question classifier. A known problem of this learner is that extra features can degrade the performance of the learner. Therefore, we performed an additional feature selection process in order to identify the best subset of features for the classification task (using bidirectional hill-climbing, see [3]).

We performed three different experiments for determining the optimal feature sets for each of the three question classes. Each of the questions in the training set was classified by training on all the other questions (leave-one-out). The best score was obtained for the coarse-grained class using four features (4, 5, 8 and 9): 85%. Identifying fine-grained classes proved to be harder: 79%, also with four features (4, 7, 8 and 9). Table classes were best predicted with three features (80%; 4, 9 and 10).

4 Multi-dimensional markup

Our system makes use of several kinds of linguistic analysis tools, including a POS tagger, a named entity recognizer and a dependency parser [7]. These tools are run offline on the entire corpus, before any questions are posed.

The output of an analysis tool takes the form of a set of annotations: the tool identifies text regions in the corpus and associates some metadata with each region. Storing these annotations as XML seems natural and enables us to access them through the powerful XQuery language. Ideally, we would like to query the combined annotations produced by several different tools at once. This is not easily accomplished, though, because the tools may produce conflicting regions. For example, a named entity may partially overlap with a phrasal constituent in such a way that



Context	Axis	Result nodes
A	select-narrow	A B C
A	select-wide	A B C E
A	reject-narrow	E D
A	reject-wide	D

Figure 2: Example document with two annotation layers.

the elements cannot be properly nested. It is thus not possible, in general, to construct a single XML tree that contains annotations from all tools.

To deal with this problem, we have developed a general framework for the representation of multi-dimensional XML markup [2]. This framework stores multiple layers of annotations referring to the same base document. Through an extension of the XQuery language, it is possible to retrieve information from several layers with a single query. This year, we migrated the corpus and all annotation layers to the new framework.

4.1 Stand-off XML

We store all annotations as stand-off XML. Textual content is stripped from the XML tree and stored separately in a *blob* file (binary large object). Two *region attributes* are added to each XML element to specify the byte offsets of its start and end position with respect to the blob file. This process can be repeated for all annotation layers, producing identical blobs but different stand-off XML markup.

Figure 2 shows an example of two annotation layers on the same text content. The stand-off markup in the first XML tree could be:

```
<A start="10" end="50">
  <B start="30" end="50"/>
</A>
```

and the second could be:

```
<E start="20" end="60">
  <C start="20" end="40"/>
  <D start="55" end="60">
</E>
```

The different markup layers are merged and stored as a single XML file. Although there will still be conflicting regions, this is no longer a problem because our approach does not require nested elements. The added region attributes provide sufficient information to reconstruct each of the annotation layers, regardless of how they are merged. In principle, we could merge the layers by simply concatenating the respective stand-off markup. However, in order to simplify query formulation, we choose to properly nest the layers down to the level of sentence elements.

4.2 Extending MonetDB/XQuery

The merged XML documents are indexed using MonetDB/XQuery [1], an XML database engine with full XQuery support. Its XQuery front-end consists of a compiler which transforms XQuery programs into a relational query language internal to MonetDB.

We extended the XQuery language by defining four new path axes that allow us to step between layers. The new axis steps relate elements by region overlap in the blob, rather than by nesting relations in the XML tree: `select-narrow` selects elements that have their region completely

contained within the context element's region; `select-wide` selects elements that have at least partial overlap with the context element's region; `reject-narrow` and `reject-wide` select the non-contained and non-overlapping region elements, respectively. The table in Figure 2 demonstrates the results of each of the new axes when applied to our example document.

In addition to the new axes, we have also added an XQuery function `so-blob($node)`. This function takes an element and returns the contents of that element's blob region. This function is necessary because all text content has been stripped from the XML markup, making it impossible to retrieve text directly from the XML tree.

The XQuery extensions were implemented by modifying the front-end of MonetDB/XQuery. An index on the offset attributes is used to make the new axis steps efficient even for large documents.

4.3 Merging annotation layers

We use a separate system, called XIRAF, to coordinate the process of automatically annotating the corpus. XIRAF combines multiple text processing tools, each having an input descriptor and a tool-specific wrapper that converts the tool output into stand-off XML annotation.

The input descriptor associated with a tool is used to select regions in the data that are candidates for processing by that tool. The descriptor may select regions on the basis of the original metadata or annotations added by other tools. For example, our sentence splitter selects document text using the `TEXT` element from original document markup. Other tools, such as the POS tagger and named-entity tagger, require separated sentences as input and thus use the output annotations of the sentence splitter by selecting `SENT` elements.

Some tools readily produce annotations in XML format, and other tools can usually be adapted to produce XML. Unfortunately, text processing tools may actually modify the input text in the course of adding annotations, which makes it non-trivial to associate the new annotations with regions in the original blob. Tools make a variety of modifications to their input text: some perform their own tokenization (i.e., inserting whitespaces or other word separators), silently skip parts of the input (e.g., syntactic parsers, when the parsing fails), or replace special symbols. For many of the available text processing tools, such possible modifications are not fully documented.

XIRAF, then, must re-align the output of the processing tools with the original blob. We have experimented with a systematic approach that computes an alignment with minimimal edit-distance. However, we found that there are special cases where a seriously incorrect alignment may have optimal edit-distance. This prompted us to replace edit-distance alignment by ad-hoc alignment rules that are specific to the textual modifications made by each tool.

The alignment of blob regions is further complicated by the use of character encodings. We use UTF-8 encoding for blob files, enabling us to represent any sequence of Unicode characters. Since UTF-8 is a variable length encoding, this means that there is no direct relation between character offsets and byte offsets. Region attributes in our stand-off markup are stored as byte offsets to allow fast retrieval from large blob files. Many text processing tools, however, assume that their input is either ASCII or Latin-1 and produce character offsets in their output. This makes it necessary for XIRAF to carefully distinguish character offsets from byte offsets and convert between them in several situations.

After conversion and re-alignment, annotations are merged into the XML database. In general, the output from a tool is attached to the element that produced the corresponding input. For example, since the POS tagger requests a sentence as input, its output is attached to the `SENT` element on which it was invoked. In principle, our use of region attributes makes the tree structure of the XML database less important. Maintaining some logical structure in the tree, however, may allow for queries that are simpler or faster.

4.4 Using multi-dimensional markup for QA

Two streams in our QA system have been adapted to work with multi-dimensional markup: the *Table Lookup* stream and *XQuesta*. The table stream relies on a set of tables that are extracted

from the corpus offline according to predefined rules. These extraction rules were rewritten as XQuery expressions and used to extract the tables from the collection text.

Previous versions of XQuesta had to generate separate XPath queries to retrieve annotations from several markup layers. Information from these layers had to be explicitly combined within the stream. Moving to multi-dimensional XQuery enables us to query several annotation layers jointly, handing off the task of combining the layers to MonetDB. Since XQuery is a superset of XPath, previously developed query patterns could still be used in addition to the new, multi-dimentional ones.

5 Probabilities

One major consequence of the multi-stream architecture of the Quartz QA system is the need for a module to choose among the candidate answers produced by the various streams. The principal challenge of such a module is making sense of the confidence scores attached by each stream to its candidate answers. This year, we made use of data from previous CLEF-QA campaigns in order to estimate correctness probabilities for candidate answers from stream confidence scores. Furthermore, we also estimated correctness probabilities conditioned on well-typedness in order to implement type-checking as Bayesian update. In the rest of this section, we describe how we estimated these probabilities. We describe how we use these probabilities in the following section.

5.1 From scores to probabilities

Each stream of our QA system attaches confidence scores to the candidate answers it produces. While these scores are intended to be comparable for answers produced by a single stream, there is no requirement that they be comparable across streams. In order to make it possible for our answer re-ranking module (described in section 6) to rank answers from different streams, we took advantage of answer patterns from previous editions of CLEF QA to estimate the probability that an answer from a given stream with a given confidence score is correct.

For each stream, we ran the stream over the questions from the previous editions of CLEF and binned the candidate answers by confidence score into 10 equally-sized bins. Then, for each bin, we used the available answer patterns to check the answers in the bin and based on these assessments, computed the maximum likelihood estimate for the probability that an answer with a score falling in the range of the bin would be correct. With these probability estimates, we can now associate with a new candidate answer a correctness probability based on its confidence score.

5.2 Type checking as Bayesian update

Type checking can be seen as a way to increase the information we have about the possible correctness of an answer. We discuss in section 6.3 how we actually type-check answers; in this section, we explain how we incorporate the results of type-checking into our probabilistic framework.

One natural way to incorporate new information into a probabilistic framework is Bayesian update. Given the prior probability of correctness for a candidate answer, $P(correct)$ (in our case, the MLE corresponding with the stream confidence score), as well as the information that it is well- or ill-typed (represented as the value of the random variable $well\_typed$), we compute $P(correct \mid well\_typed)$, the updated probability of correctness given well- (or ill-)typedness as follows:

$$\begin{aligned} P(correct \mid well\_typed) &= \frac{P(correct \wedge well\_typed)}{P(well\_typed)} \\ &= \frac{P(correct) \times P(well\_typed \mid correct)}{P(well\_typed)} \\ &= P(correct) \times \frac{P(well\_typed \mid correct)}{P(well\_typed)} \end{aligned}$$

In other words, the updated correctness probability of a candidate answer, given the information that it is well- or ill-typed, is the product of the prior probability and the ratio $\frac{P(well\_typed \mid correct)}{P(well\_typed)}$.

We estimate the required possibilities by running our type-checker on assessed answers from CLEF-QA 2003, 2004, and 2005. For question types in which type-checking is actually possible, the ratio for well-typed answers is 1.25:

$$\frac{P(well\_typed = \text{True} \mid correct = \text{True})}{P(well\_typed = \text{True})} = 1.25$$

The ratio for ill-typed answers is 0.34:

$$\frac{P(well\_typed = \text{False} \mid correct = \text{True})}{P(well\_typed = \text{False})} = 0.34$$

6 Answer processing

The multi-stream architecture of Quartz embodies a high-recall approach to question answering—the expectation is that using a variety of methods to find a large number of candidate answers should lead to a greater chance of finding correct answers. The challenge, though, is choosing *correct* answers from the many candidate answers returned by the various streams. The answer processing module described in this section is responsible for this task.

6.1 High-level overview

The algorithm used by the Quartz answer processing module is given here as Algorithm 1.

Algorithm 1 High-level answer processing algorithm

1: **procedure** AnswerProcessing($candidates, question, tc$)
2: $clusters \leftarrow$ Cluster($candidates$)
3: **for** $cluster \in clusters$ **do**
4: **for** $ans \in cluster$ **do**
5: $ans.well\_formed$, $ans.well\_typed$, $ans.string \leftarrow$ FormTypeCheck($ans, question$)
6: **end for**
7:
$$P(cluster) = 1 - \prod_{ans \in cluster} \left(1 - P(ans) \times \frac{P(well\_typed = ans.well\_typed \mid ans)}{P(well\_typed = ans.well\_typed)}\right)$$
8: $cluster.rep\_answer \leftarrow \text{argmax}_{ans \in \{\ a \mid a \in cluster \wedge a.well\_formed\ \}} \text{length}(ans)$
9: **end for**
10: $ranked\_clusters \leftarrow \text{sort}_P(clusters)$
11: **end procedure**

First, candidate answers for a given question are clustered (line 2: section 6.2). Then, each cluster is evaluated in turn (lines 3–9). Each answer in the cluster is checked for well-formedness and well-typedness (line 5: section 6.3): $ans.well\_formed$ is a boolean value indicating whether some part of the original answer is well-formed ($ans.string$ is the corresponding part of the original answer); $ans.well\_typed$ is a boolean value indicating whether the answer is well-typed. The correctness probability of the cluster $P(cluster)$ (line 7) is the combination of the updated correctness probabilities of the answers in the cluster (the prior correctness probabilities $P(ans)$ updated with type-checking information, as described in section 5.2). (Note that for one of the runs we submitted for the 2006 CLEF QA evaluation we turned off type-checking, so that $P(cluster)$ was simply $1 - \prod_{ans}(1 - P(ans))$.) The longer of the well-formed answers in the cluster is then chosen as the representative answer for the cluster (line 8). Finally, the clusters are sorted according to their correctness probabilities (line 10).

6.2 Answer clustering

Answer candidates with similar or identical answer strings are merged into clusters. In previous versions of our system, this was done by repeatedly merging pairs of similar answers until no more similar pairs could be found. After each merge, the longest of the two answer strings was selected and used for further similarity computations.

This year, we moved to a graph-based clustering method. Formulating answer merging as a graph clustering problem has the advantage that it better captures the non-transitive nature of answer similarity. For example, it may be the case that the answers *oorlog* and *Wereldoorlog* should be considered similar, as well as *Wereldoorlog* and *wereldbeeld*, but not *oorlog* and *wereldbeeld*. To determine which of these answers should be clustered together, it may be necessary to take into account similarity relations with the rest of the answers.

Our clustering method operates on a matrix that contains a similarity score for each pair of answers. The similarity score is an inverse exponential function of the edit distance between the strings, normalized by the sum of the string lengths. The number of clusters is not set in advance but is determined by the algorithm. We used an existing implementation of a spectral clustering algorithm [5] to compute clusters within the similarity graph. The algorithm starts by putting all answers in a single cluster, then recursively splits clusters according to spectral analysis of the similarity matrix. Splitting stops when any further split would produce a pair of clusters for which the normalized similarity degree exceeds a certain threshold. The granularity of the clusters can be controlled by changing the threshold value and the parameters of the similarity function.

6.3 Checking individual answers

The typing and well-formedness checks performed on answers depends primarily on the expected answer type of the question. Real type-checking can only be performed on questions whose expected answer type is a named-entity, a date, or a numeric expression. For other questions, only basic well-formedness checks are performed. Note that the probability update ratios described in section 5.2 are only computed on the basis of answers that are type-checkable. For answers to other questions, we use the same ratio for ill-formed answers as we do for ill-typed answers (0.34), but we do not compute any update for well-formed answers (i.e., we update with a ratio of 1.0).

For answers to all questions, two basic checks are performed. If an answer consists solely of non-alphanumeric characters or is wholly contained as a substring of the question, it is immediately rejected as ill-formed and ill-typed. For answers to questions with the following answer types only the very basic well-formedness checks noted are additionally performed:

- Abbreviation, Units: answer must be a single word with some alphabetic characters
- Expansion, Definition, Manner, Term: answer must contain some alphabetic characters

For questions expecting named entities, dates, or numeric answers, more significant well-formedness and well-typedness checks are performed. For numeric answers, we check that the answer consists of a number (either in digits or spelled out) with optional modifiers (e.g., *ongeveer*, *meer dan*, etc.) and units. For names and dates, we look at the justification snippet and run an NE-tagger or date tagger on the snippet, as appropriate. In addition to verifying that the candidate answer is a name (for well-formedness) of the correct type (for well-typedness), we also check whether there are additional non-name words at the edges of the answer and remove them, if necessary. The results of answer checking are then boolean values for well-formedness and well-typedness, as well as the possibly edited answer string.

7 Runs

We submitted two Dutch monolingual runs. The run uams06Tnlnl used the full system with all streams and final answer selection. The run uams06Nnlnl used the full system but without type-checking.

Run	Total	Right	Unsupported	Inexact	Wrong	% Correct
uams06Tnlnl	200	40	2	4	154	20%
uams06Nnlnl	200	41	3	4	152	21%

Table 1: Assessment counts for the 200 top answers in the two Amsterdam runs submitted for Dutch monolingual Question Answering (NLNL) in CLEF-2006. In both runs, about 20% of the questions were answered correctly.

Question type	Total	Right	Unsupported	Inexact	Wrong	% Correct
factoid	116	28	1	2	85	24%
definition	39	9	0	1	29	23%
temporally restricted	32	3	1	1	27	8%
non-list	187	40	2	4	141	21%
list	13	0	6	0	31	0%

Table 2: Assessment counts per question type for the 200 questions in the Amsterdam run with type checking (uams06Tnlnl). We have regarded all questions with time expressions as temporally restricted rather than the single one mentioned in the official results. Note that for the 13 list questions there were more than 13 answers since list questions allowed a maximum of five answers.

8 Results

The question classifier performed as expected for coarse question classes: 86% correct compared with a score of 85% on the training data. For most of the classes, precision and recall scores were higher than 80%; the exceptions are `miscellaneous` and `number`. For assigning table classes, the score was much lower than for the training data: 56% compared with 80%. We did not evaluate fine-grained class assignment because these classes are not used in the current version of the system.

Table 1 lists the assessment counts for the two University of Amsterdam runs for the question answering track of CLEF-2006. The two runs had 14 different top answers of which four were assessed differently. In 2005 our two runs contained a large number of inexact answers (28 and 29). We are happy about those numbers being lower in 2006 (4 and 4) and the most frequent problem, extraneous information added to a correct answer, has almost disappeared. However, the number of correct answers dropped as well, from 88 in 2005 to 41. This is at least partly caused by the fact that the questions were more difficult this year.

Like last year, the questions could be divided in two broad categories: questions asking for lists and questions requiring singular answers. The second category can be divided in three subcategories: questions asking for factoids, questions asking for definitions and temporally restricted questions. We have examined the uams06Tnlnl-run answers to the questions of the different categories in more detail (Table 2). Our system failed to generate any correct answers for the list questions. For the non-list questions, 21% of the top answers were correct. Within this group, the temporally restricted questions[1] (8% correct) posed the biggest challenge to our system. In 2005, we saw similar differences between factoid, definition and temporally restricted questions. At that time the difference between the last category and the first two could be explained by the presence of a significant number of incorrect answers which would have been correct in another time period. This year, no such answers were produced by our system.

More than 75% of our answers were incorrect. We examined the answers given to the first twelve questions in more detail in order to find out what the most important problems were (see appendix A). The top answer to the first question is *leak in the tank of space shuttle*. It is unclear to us why the correct and more frequent apposition *space shuttle* was not selected as the answer.

[1]The official assessments mention the presence of only one temporally restricted question, which is very unlikely. We have regarded all questions with time expressions (32) as temporally restricted.

For question two, three of the top five answers are correct, but the top answer is wrong. It consists of a long clause which seems to have been identified as an apposition. The third question has been answered correctly. All top five answers are short.

No correct answer was found for question four because the date in the question was wrong (should have been 1938). Question five was answered correctly (NIL). For question six, the top answer is inexact because it only mentions a city and not the name of the concert hall. The other four answers are poorly motivated by the associated snippets. The same is true for all the answers generated for question seven. Here, the correct answer is NIL. Four of the five answers produced for question eight do not have the correct type. They should have been a time period, but apart from the top answer, they consist of arbitrary numeric expressions.

Question nine is hard to answer. We have not been able to find the required answer (one million) in the collection. Our system generated five numbers which had nothing to do with the required topic. Two of the numbers were clearly wrong: *000*. The top answer for question ten has an incorrect type as well (`noun` rather than the required `organization`). No answers were generated for questions eleven and twelve. The first required a search term expansion (from UN to United Nations). The NIL answer for the second is a surprise. The question phrases *foreign*, *movie* and *Oscar* can be found back in the collection close the correct answers which have the required type.

Based on this analysis, we have created the following list of future work:

- As the answers to the first two questions show, our system still prefers infrequent long answers over frequent short ones. This often leads to highly ranked incorrect answers. We should change the ranking part of the system and give a higher weight to frequency rather than string length.
- Another way to decrease the number of incorrect long answers is to prevent them from appearing in the first place. This means changing our information extraction modules.
- One part of our architecture which is currently poorly understood is the part that generates queries. In our analysis, we found several cases of undergeneration and overgeneration of answers. We suspect that the query generation component is involved in the generation of at least some of these answers.
- Although part of our work has been focused on matching types of answers with question types, it seems that our current system still has problems in this area (questions nine and ten). More work is required here.

In the last four years, we have submitted multiple runs with minor differences. Given our current scoring level it would be interesting to aim at runs that are more different. This could have a positive effect on our standing in the evaluation.

9 Conclusion

We have described the fourth iteration of our system for the CLEF Question Answering Dutch mono-lingual track (2006). This year, our work has focused on converting all data repositories of the system (text, annotation and tables) to XML and allowing them to be accessed via the same interface. Additionally, we have modified parts of our system which we had suspected of weaknesses: question classification and answer processing.

At this point, we would like to know if the modifications of the system have resulted in improvemed performance. The system's performance on this year's questions (20% correct) was much worse than in 2005 (45%). However, the 2006 questions were more difficult than those of last year, and we expect that the workshop will show that the performance of other participants has also dropped. It would be nice to be able to run last year's system on the 2006 questions but unfortunately, we do not have a copy of that system available. Applying the current system to

last year's questions is something we can and should do, but the outcome of that experiment may not be completely reliable since those questions have been used for tuning the system.

At this moment, we simply do not know if our work has resulted in a better system. One thing we know for certain: there is a lot of room for improvement. Obtaining 20% accuracy on factoid questions is still far away from the 70% scores obtained by the best systems participating in TREC. Luckily, our 2006 CLEF-QA participation has identified key topics for future work: information extraction, query generation, answer type checking, and answer ranking. We hope that work on these topics will lead to better performance in 2007.

10 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.

References

[1] MonetDB. Website: `http://www.monetdb.nl/`.

[2] Wouter Alink, Valentin Jijkoun, David Ahn, Maarten de Rijke, Peter Bonz, and Arjen de Vries. Representing and querying multi-dimensional markup for question answering. In *Proceedings of the 5th Workshop on NLP and XML*, pages 3–9. ACL, 2006.

[3] Rich Caruana and Dayne Freitag. Greedy attribute selection. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 28–36. New Brunswick, NJ, USA, Morgan Kaufman, 1994.

[4] 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/.

[5] Luigi Dragone. Spectral clusterer for WEKA, 2002.
Webpage: `http://www.luigidragone.com/datamining/spectral-clustering.html`.

[6] 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.

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

[8] Pik Vossen, editor. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, 1998.

A Example questions and answers

This appendix contains the top twelve questions and answers of the monolingual (NLNL) run uams06Tnlnl. The assessed score can be found before each answer: Right, Unsupported, ineXact or Wrong.

1. 0001 Wat is Atlantis?

(a) W lek in een heliumtank van het ruimteveer

(b) W na zijn historische missie naar het Russische ruimtestation Mir veilig teruggekeerd op aarde

(c) W gisteren met succes vastgekoppeld aan het Russische ruimtestation Mir

(d) W gisteren om 15

(e) W een doorzichtige allegorie op het fascisme en het fascistische dedain voor het menselijke leven

2. 0002 Wat is Hubble?

(a) W genoemd naar de Amerikaanse astronoom Edwin Hubble , die deze waarde in 1929 voor het eerst bepaalde

(b) R de Hubble-ruimtetelescoop

(c) W er oorzaak van dat het heelal nog nooit zo raadselachtig is geweest als nu

(d) R een sterrenkijker die we per raket de ruimte in hebben geschoten , vanwaar hij nu plaatjes naar de

(e) R de constante

3. 0003 Wat is Lusa?

(a) R Portugese persbureau

(b) W Het kabinet

(c) W 530 kilometer

(d) W AP

(e) W hulp

4. 0004 Waarheen verhuisde Freud in 1939?

(a) W NIL

5. 0005 Waar is de olie-tanker Prestige gezonken?

(a) R NIL

6. 0006 Waar vindt de uitreiking van de Oscars plaats?

(a) W Los Angeles

(b) W Rotterdam

(c) W Nederland

(d) W Hollywood

(e) W Philadelphia

7. 0007 Hoeveel ruimtevaartuigen werden er gelanceerd in 1977?

(a) W vijfentwintigduizend

(b) W elf ton

8. 0008 Hoe lang duurt de ruimtemissie Voyager?

(a) W 1655

(b) W zeven

(c) W twee ton

(d) W drie

(e) W 5150 km

9. 0009 Hoeveel illegale immigranten zijn er nu in de EU landen?

   (a) W 51 procent

   (b) W 42 procent

   (c) W 000

   (d) W 000 dollar

   (e) W twaalf

10. 0010 Welke partij heeft Karadzic opgericht?

   (a) W partijen

   (b) W AFP

11. 0011 Welke verklaring werd in 1948 door de VN aangenomen?

   (a) W NIL

12. 0012 Welke film heeft de Oscar voor de beste buitenlandse film gekregen?

   (a) W NIL