THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 9, Number 3/2008, pp. 000–000

IMPORTANT PRACTICAL ASPECTS OF AN OPEN-DOMAIN QA SYSTEM PROTOTYPING

Radu ION, Dan ŞTEFĂNESCU, Alexandru CEAUŞU

Research Institute for Artificial Intelligence, Romanian Academy
13, "Calea 13 Septembrie", Bucharest 050711, Romania
Corresponding author: Radu ION, radu@racai.ro

This paper deals with several practical aspects of an open-domain QA system implementation. Specifically, we describe the process of the query formulation from the question analysis as well as the identification and the ordering of the text snippets that may contain the sought answer in the context of the CLEF 2008 Romanian-Romanian QA track. We show that the tuning of these system components is important with respect to the output of the QA system and will directly affect the performance measures.

Key words: QA systems; query formulation; term relevance; snippet ranking; CLEF.

1. INTRODUCTION

Open-domain question answering systems (QA systems) aim at producing an exact answer to a natural language formulated question using a dynamic collection of documents. The "dynamic" attribute specifies that the collection is not fixed thus being possible to modify it. One can add documents to it, even from different domains or can delete documents. This mutable nature of the document collection imposes a rather hard to obey constraint on the construction of the QA system which is the ability of applying *a general processing flow* from question analysis to answer extraction to produce natural language questions answers.

Typically, the general processing flow of an open-domain QA system involves the following steps:

1. question analysis in which the topic/focus articulation, question type and answer type are identified and in which the query formulation (the translation from natural language to the search engine syntax) takes place;
2. information retrieval in which the top N documents matching the query from the previous step are returned;
3. answer extraction in which a set of relevant snippets are extracted from the documents in the previous step and, if an exact answer (the minimal, grammatical substring that completely answers the user's question) is required, this answer is searched in the collection of relevant snippets.

The most difficult operation is the answer extraction especially if an exact answer is required. Nowadays, this is the task of a standard QA system but the exact answer requirement was not mandatory with the older QA TREC (http://trec.nist.gov/) competitions in which only a 50 byte snippet was expected. It is fairly obvious that errors will propagate from step 1 to step 3 and that a defective question analysis will result in an erroneous query that in turn, when used by the search engine, will fail to retrieve the relevant documents. Needless to say that the correct snippets cannot be identified and with them, the correct answers.

The best (and probably the most sophisticated) QA systems to date [3, 2] employ answer validation which essentially is a logical computation describing the fact that a string is a correct answer to the question. This procedure naturally requires a passage retrieval (PR) and ranking module which, unfortunately, is not described at all (aside from the fact that in the 2006 paper, the PR ranks passages based on "lexical similarity"). The recent literature on QA systems seems to take the same direction focusing on the answer extraction procedure (naturally, since in the most interesting part) and only briefly describing the query formulation and passage (or snippet) selection and ranking for answer extraction.

The present paper will focus exactly on these two components of an open-domain QA system which, in our opinion, when tuned to the best of their abilities, will facilitate any method of answer extraction. Thus the next chapter will explain our indexing and searching engine and after that, we will present a method of query formulation and evaluate it on the set of questions from CLEF 2008 Romanian-Romanian QA track. Finally, we will present a method of selection and ranking of snippets and present MRR scores of this method on the same set of questions.

2. THE SEARCH ENGINE

The document collection that we use to test our system is that of the CLEF 2008 Romanian-Romanian track competition (http://www.clef-campaign.org/). This collection is composed of 43486 Romanian language documents from Wikipedia (http://ro.wikipedia.org/). Each document has a title and several sections made up from paragraphs. All the logical sections of the documents were preprocessed with the TTL module [1] to obtain POS tagging, lemmatization and chunking of the text within.

The search engine is a C# port of the Apache Lucene (http://lucene.apache.org/) full-text searching engine. Lucene is a Java-based open source toolkit for text indexing and Boolean searching allowing queries formed with the usual Boolean operators such as AND, OR and NOT. Furthermore, it is capable to search for phrases (terms separated by spaces and enclosed in double quotes) and also to allow boosting for certain terms (the importance of a term is increased with the caret character '^' followed by an integer specifying the boost factor). We also used the field-specific term designation: a term may be prefixed by the field name to constrain the search to specific fields (such as title or text for instance) in the document index.

For the construction of the index, we considered that every document and every section within a document have different fields for the surface form of the words and their corresponding lemmas. This kind of distinction applies to titles and paragraph text resulting in four different index fields: title word form (**`title`**), title lemma (**`ltitle`**), paragraph word form (**`text`**) and paragraph lemma (**`ltext`**). We used the sentence and chunk annotation (from the TTL output) to insert phrase boundaries into our term index: a phrase query cannot match across different chunks or sentences. Thus, for instance, if we want to retrieve all documents about the TV series Twin Peaks, we would first like to search for the phrase "Twin Peaks" in the title field of the index (Lucene syntax **`ltitle`**`:"Twin Peaks"`) and then, to increase our chance of obtaining some hits, to search in the word form field of the index for the same phrase (Lucene syntax **`text`**`:"Twin Peaks"`). Consequently, this Lucene query would look like this: **`ltitle`**`:"Twin Peaks"` **`OR`** **`text`**`:"Twin Peaks"`.

The result of searching is a list of documents which is ordered by the relevance score of each document with respect to the posed query. Within each document, if a condition applies (see below), its sections are also ranked and listed in decreasing order with respect to the relevance to the query. We have used Lucene's default scoring formulas when ranking documents and their sections.

Document retrieval is done with a special technique that allows for automatic query relaxation. Given a Boolean query that is a conjunction of n terms, the system will first try to match all of the query terms (which may be prefixed by different fields) against the document index. If the search is unsuccessful, the system will try to exhaustively match $n - k$ ($1 \leq k < n$) of the query terms until a relaxed query (which now contains $m < n$ terms) returns at least one document from the document index. If the query is not a conjunction of terms (has a more complicated structure involving grouping and usage of other Boolean operators) then it is submitted as is and the results are returned to the application. It is the application's responsibility to deal with too specific queries that have a void result.

If the query is a conjunction of terms, the search engine also provides section ranking within each of the returned documents. This is achieved by automatically performing a second search within these documents using the section index and a new query. The new query is composed by joining the terms from the relaxed query that produced the document list with the disjunction operator (OR).

3. QUERY FORMULATION

Our query formulation strategy improves the one described in [7] which was successfully used in the CLEF 2007 Romanian-Romanian QA track. The input question must first be preprocessed with the TTL module to obtain POS tagging, lemmatization and chunking information.

Query formulation from a POS tagged, lemmatized and chunked input question basically constructs a conjunction of terms which are extracted from the chunking information of the input sentence. Specifically, the algorithm considers all the noun phrases of the question from which it constructs terms. Each term is prefixed by a text or title field and is present both in lemma and word form.

The CLEF 2007 version of the algorithm used to take into account all the word boundary substrings of each noun phrase regardless of their likeliness to appear. For instance, for the noun phrase "*cele mai avansate tehnologii ale armatei americane*"/"*the most advanced technologies of the US army"*, terms like "*mai avansate tehnologii ale*" or "*cele mai avansate*" were valid. The present version of the question formulation algorithm fixes this aberration by constraining the substrings to be proper noun phrases themselves. In addition to that, the assignment of fields to each term was revised. Following, is the summary of modifications:

1. substring starting or ending with words of certain parts of speech are not considered terms; for instance substrings ending with adjectives or articles or beginning with adverbs;
2. substrings that do not contain a noun, a numeral or an abbreviation are not considered terms;
3. substrings starting with words other than nouns, numerals or abbreviations are not to be searched in the title field;
4. single word terms in occurrence form are not to be searched in the title field.

The improvement of the query formulation algorithm is exemplified for the question "*Cine erau părinții lui Ares, conform mitologiei grecești?*"/"*Who were Ares' parents, according to the Greek mythology?*". The older version of the query formulation produced the query (the space between terms signifies the existence of the AND operator)

```
ltitle:"părinte lui Ares" ltext:"părinte lui Ares" ltext:"lui Ares" ltext:"părinte lui"
ltitle:"mitologie grecesc" ltext:"mitologie grecesc" ltitle:mitologie ltext:mitologie
ltext:grecesc ltitle:părinte ltext:părinte ltitle:Ares ltext:Ares title:"părinții lui
ares" text:"părinții lui ares" text:"lui Ares" text:"părinții lui" title:"mitologiei
grecești" text:"mitologiei grecești" title:mitologiei text:mitologiei text:grecești
title:părinții text:părinții title:ares text:ares
```

while the new version of the query after applying steps 1 – 4 is the following

```
ltitle:"părinte lui Ares" ltext:"părinte lui Ares" ltitle:"mitologie grecesc"
ltext:"mitologie grecesc" ltitle:părinte ltext:părinte ltitle:Ares ltext:Ares
ltitle:mitologie ltext:mitologie title:"părinții lui ares" text:"părinții lui ares"
title:"mitologiei grecești" text:"mitologiei grecești" text:părinții text:ares
text:mitologiei .
```

Thus, we can see that from a query of 26 terms, we ended up with a smaller query of 17 terms. From the 26 initial terms, improper and unlikely to appear terms such as `ltext:"lui Ares"`, `ltext:"părinte lui"`, `text:"lui Ares"`, `text:"părinții lui"` and so on have been eliminated.

The importance of the single word heuristic for title searching (step 4) becomes clearer in the light of a question like "*Ce este Pământul?*"/ "*What is the Earth?*". The queries generated by the two versions (older (a) and present (b)) are: (a) `ltitle:pământ ltext:pământ title:pământul text:pământul` and (b) `ltitle:pământ ltext:pământ text:pământul`. The older query retrieved as the top document a document with the title "*Pământul de Mijloc*"/"*Middle Earth*" referring to a fictional land created by J.R.R. Tolkien because of the existence of the term `title:pământul` and because of the "match all if you can" heuristic of the search engine which matched both the word form and the lemma of the term in the title field. With the second query, all the terms were matched as in the previous case but the top scored document now refers to the planet Earth (Middle Earth document being listed below probably because its title is longer than the title of the planet Earth document which is "*Pământ*").

We wanted to evaluate the MRR score of this query formulation algorithm onto the 200 question test set of the CLEF 2008 Romanian-Romanian QA track in which we participated this year. The track proposed groups of questions to be automatically answered. All questions in a group were topically related and the first question in a group was anaphora free. Subsequent questions in the same group could contain anaphoric references with the referents being either the answer to a previous question or the topic of a previous question.

Our system aims, for the time being, at answering unrelated, anaphora free questions. To this end, we used a "normalized" version of the 200 question test set in which we have manually resolved all anaphoric references in a question group.

Table 1: Query formulation algorithm improvements

	MRR	**Coverage**
Initial	~0.7000	0.7500
Step 1	0.7366	0.7624
Step 2	0.7455	0.7715
Step 3	0.7689	0.8012
Step 4	0.7776	0.8092

The query structure (its terms in our case) directly influences the *accuracy* and the *coverage* of the search engine. The accuracy is computed as a *Mean Reciprocal Rank* (MRR) score for documents (Voorhees, 1999), while the coverage is practically the recall score (coverage is the upper bound for the MRR). Although we primarily aim for covering all the questions (which means that we want to relax the queries in order to get documents/sections containing answers for as many questions as possible), a good MRR will ensure that these documents/sections will be among the top returned. Consequently, the detection of the exact answer should be facilitated if this procedure considers the ranks assigned by the search engine to the returned documents. The greater the MRR score, the better the improvement.

As Table 1 shows, starting from a MRR of around 0.7 and a coverage of 0.75 obtained with the 2007 version of the query formulation algorithm, the improved query formulation algorithms now achieves a MRR of 0.7776 and a coverage of 0.8092. The figures were computed using the reference Gold Standard of the CLEF 2008 Romanian-Romanian QA track in which for each question, the document identifier of the document containing the right answer is listed. We have not considered the questions which had a NIL answer and as such, no document identifier assigned.

The implementation of this query formulation algorithm is a web service (the WSDL description of which can be found at http://shadow.racai.ro/QADWebService/Service.asmx?WSDL) that takes the Romanian question as input and returns the query. To obtain POS tagging, lemma and chunking information, the web service uses another web service, TTL [6].

4. SNIPPET SELECTION AND RANKING

The snippet (or passage) selection is important to the answer extraction procedure because the answers should be sought only into small fragments of text that are relevant with respect to the query and implicitly to the question. This is because, usually, the answer extraction procedure is computationally expensive (different reasoning mechanisms, parsing, ontology interrogation, etc. are involved) and thus, would be best to apply it onto small fragments of text.

Snippet selection uses the question analysis to identify passages of text that, potentially, contain the answer to the question. The question analysis produces the focus and the topic of the question and was described in [7]. Basically, it uses the linkage of the question obtained with LexPar [1] to identify linking patterns that describe the syntactic configuration of the focus, the main verb and the topic of the question. For instance, in the question "*Câte zile avea aprilie înainte de 700 î.Hr.?*"/" *How many days did April have before 700 B.C.?*" the focus is the word "*zile*"/"*days*" and the topic is "*aprilie*"/"*April*" because the linkage of this question contains the links interrogative determiner "*Câte*"/"*How many*" – noun "*zile*"/"*days*", noun "*zile*"/"*days*" – main verb "*avea*"/"*did have*" and main verb "*avea*"/"*did have*" – noun "*aprilie*"/"*April*" which determine a syntactic pattern in which the focus is the first noun and the topic, the second noun.

For each section of each returned document, the snippet selection algorithm considers windows of at most *N* words at sentence boundary (that is, no window may have fewer than *N* words but it may have more than *N* words to enforce the sentence boundary condition). Each window is scored as follows (each word is searched in its lemma form):

- if the focus of the question is present, add 1;
- if the topic of the question is present, add 1;
- if both the topic and the focus of the question are present, add 10;
- if the *k*-th dependant of the focus/topic of the question is present add $1 / (k + 1)$. The *k*-th dependent of a word *a* in the linkage of a sentence is the word *b*, if there exists a path of length *k* between *a* and *b*.

The above heuristics are simple and intuitive. Each window in which either focus or topic are found, receives one point. If both are to be found, a 10 point bonus is added because the window may contain the reformulation of the question into a statement which will thus resolve the value of the focus. The last heuristic aims at increasing the score of a window which contains dependents of the focus and/or topic but with a value which decreases with the distance (in links) between the two words. The selection algorithm will retrieve at most *M* top-scored snippets from the documents returned by the search engine. A snippet may be added only if its selection score is greater than 0.

The snippet selection algorithm provides an initial ranking of the snippets. However, there are cases when the focus/topic is not present in its literal form but in a semantically related form like a synonym, hypernym, etc. This problem is known as the "lexical gap" between the question formulation and the text materialization of the answer. In these cases, our snippet selection procedure will assign lower scores to some of the important snippets because it will not find the literal representation of the words it looks for. To lighten the impact of this problem onto the snippets' scores, we developed a second ranking method which uses *lexical chains* to score the semantic relatedness of two different words.

The term "lexical chain" refers to a set of words which, in a given context (sentence, paragraph, section and so on), are semantically related to each other and are all bound to a specific topic. For instance, words like "*tennis*", "*ball*", "*net*", "*racket*", "*court*" all may form a lexical chain if it happens that a paragraph in a text contains all of them. Moldovan and Novischi [4] used an extended version of the Princeton WordNet (http://wordnet.princeton.edu/) to derive lexical chains between the meanings of two words by finding semantic relation paths in the WordNet hierarchy. Thus, a lexical chain is not simply a set of topically related words but becomes a path of meanings in the WordNet hierarchy.

Following [4] we have developed a lexical chaining algorithm using the Romanian WordNet [5] that for two words in lemma form along with their POS tags returns a list of lexical chains that exist between the meanings of the words in the Romanian WordNet. Each lexical chain is scored as to the type of semantic relations it contains. For instance, the synonymy relation receives a score of 1 and a hypernymy/hyponymy relation, a score of 0.85. Intuitively, if two words are synonymous, then their semantic similarity measure should have the highest value. The score of a lexical chain is obtained by summing the scores of the semantic relations that define it and dividing the sum to the number of semantic relations in the chain. All the semantic relations present in Romanian WordNet have been assigned scores (inspired by those in [4]) between 0 and 1. Thus, the final score of a lexical chain may not exceed 1.

Our lexical chaining algorithm differs from the one in [4] in several aspects. Firstly, Romanian WordNet does not have word sense disambiguated glosses and as such, we had to consider all the senses of a gloss literal. Secondly, our algorithm expands the "semantic frontier" of a meaning more than once (it has a parameter which has been experimentally set to 5) allowing for discovery of "deeper" lexical chains.

Using the lexical chaining mechanism, we were able to re-rank the snippets that were selected with the previous procedure by computing the best lexical chain scores between focus, topic and their dependents and the words of the window. We have not computed lexical chains for words that were identical because this case is successfully covered by the snippet selection mechanism. Thus, for instance, for the question "*Câți oameni încap în Biserica Neagră din Braşov?*"/"*How many people fit into the Black Church in Braşov?*" the text snippet "*Biserica Neagră este cel mai mare edificiu de cult în stil gotic din sud-estul Europei, măsurând 89 de metri lungime şi 38 de metri lățime. În această biserică încap circa 5.000 de persoane.*" was able to receive a higher score due to the fact that the question focus "*oameni*" is materialized as "*persoane*" (they are synonymous).

As with the queries, we wanted to evaluate the two methods of snippet ranking individually and in combination. We have set *N* (the number of words in a snippet) to 10 and 50 (these settings for *N* roughly correspond to 50-byte and 250-byte runs of the previous TREC QA competitions) and *M* (the number of retrieved snippets) to 20. We have also considered only the top 10 documents returned by the search engine. The question test set/Gold Standard are the same as before and we counted a snippet (MRR style) only if it contains the answer *exactly* as it appears in the Gold Standard (no interest in NILs). Table 2 summarizes the results.

Table 2: MRR performance of the snippet selection and ranking algorithm

N	Key word ranking	Lexical chain ranking	The combination	Coverage
10	**0.4372**	0.3371	0.4037	0.6894
50	0.4589	0.3679	**0.4747**	0.7

The combination of the two ranking methods consists in simply adding the scores provided for each snippet. When the snippet contains 10 words, the lexical chaining ranking method does not help the key word ranking method because the semantic relatedness evidence is reduced by the short size of the snippet. When the snippet size increases (50 words), the contribution of the lexical chaining is more significant and this is reflected in the MRR score.

5. CONCLUSIONS

We have presented a QA system setup which focuses on several details that are likely to improve the answer extraction algorithm. Firstly, we have shown that an improved query generation procedure will boost the MRR of the documents that are retrieved with direct implications on the discovery of the right answer (we believe that this aspect has not yet received enough attention from the QA community). Secondly, we have proposed a method of snippet selection with an acceptable MRR but with a promising coverage. If the answer extraction procedure makes use of the ranking of the retrieved snippets, its performances should have the snippet's MRR as a baseline.

The first improvement to the query formulation algorithm is to use some kind of query expansion (a technique which, for the moment, we have not implemented). Query expansion is partially supplemented by the lexical chaining procedure but when no question key words are present in the indexed text, the present version of the query formulation algorithms fails to generate a query that will have a result.

We intend to transform all the modules of this QA system into web services and to construct an open-domain QA system that can be easily upgraded with new components such as (in the immediate future) an answer extraction algorithm. So far, the QA system works for Romanian but it can easily be adapted to other languages supported by TTL for which suitable wordnets are available.

REFERENCES

1. ION, R., *Word Sense Disambiguation Methods Applied to English and Romanian*, PhD thesis, Romanian Academy, Bucharest, 2007.
2. MOLDOVAN, D., BOWDEN, M., TATU, M., *A Temporally-Enhanced PowerAnswer in TREC 2006*, Proceedings of the Text Retrieval Conference (TREC-15), Gaithersburg, Maryland, 2006.
3. MOLDOVAN, D., CLARK, CH., BOWDEN, M., *Lymba's PowerAnswer 4 in TREC 2007*, Proceedings of the Text Retrieval Conference (TREC-16), Gaithersburg, Maryland, 2007.
4. MOLDOVAN, D., NOVISCHI, A., *Lexical Chains for Question Answering*, Proceedings of COLING 2002, Taipei, Taiwan, pp 674 – 680, 2002.
5. TUFIŞ, D., ION, R., BOZIANU, L., CEAUŞU, AL., ŞTEFĂNESCU, D., *Romanian WordNet: Current State, New Applications and Prospects*, In Attila Tanács et al. (eds.): "Proceedings of the Fourth Global WordNet Conference (GWC 2008)", Szeged, Hungary, pp 441 – 452, 2008.
6. TUFIŞ, D., ION, R., CEAUŞU, AL., ŞTEFĂNESCU, D., *RACAI's Linguistic Web Services*, Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), Marrakech, Morocco, 2008.
7. TUFIŞ, D., ŞTEFĂNESCU, D., ION, R., CEAUŞU, AL., *RACAI's Question Answering System at QA@CLEF2007*, In Carol Peters et al. (eds.): "Evaluation of Multilingual and Multi-modal Information Retrieval 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, Revised Selected Papers", Lecture Notes in Computer Science, Springer-Verlag, (in press).

Received July 18, 2008