The Encyclopedia Expert

Zdeněk Svoboda

April 5, 2002

Abstract

This paper describes the architecture and implementation of an open-domain question answering (QA) engine developed for the Czech language, using a general encyclopedia as the data source. Techniques from both the knowledge base systems and information retrieval systems are covered. Finally the evaluation results are presented.

1 Introduction

It is a well known fact that the amount of information in the society is rapidly growing. Besides the possitive impacts on our lives, this involves problems, such as information overload and low retrieval efficiency. The only way to address this problem is to develop sophisticated methods of information storage and natural methods of information retrieval.

One of the most natural methods of information retrieval is the question answering (*QA*). It became more popular in the last years and many English speaking *QA* engines were developed. However there was no complex open-domain *QA* engine working with the Czech language until recently. The architecture of our engine was partially inspired by the general design of the *QA* engines, presented at the *TREC 8* conference, particularly by the *Lasso* engine [1] and the *AT&T* engine [2]. This system was developed as a master thesis on the Faculty of Informatics in Brno in the year 2001. The original text of the thesis [3] (only in Czech) provides implementation details not covered by this paper due to the space limit.

2 Preprocessing the text data

The first step to be taken was to bring structure to the original raw text data file and to find a suitable method of indexing it for the purpose of *QA*. The encyclopedia was split to individual entries (and their meanings) and then we applied number of heuristic decision rules to identify the text fragments semantically corresponding to the entities *MAN*, *COUNTRY*, *RIVER*, *MOUNTAIN*, *CITY* and some of their attributes such as *name*, *born*, *population*, *location*, etc. This structure was captured using the *XML* markup language and the suitable *DTD*.

The resulting tagged text was indexed in two manners. The first one borrows some features of the knowledge base *QA* systems. The second one makes use of the traditional vector space text indexing and retrieving concept.

3 General overview of the engine architecture

The input question proceeds sequentially through the chain of the engine modules. The interface between modules is defined by *XML* tagging of the question. Now we describe the three main modules of the system — the taxonomy, the *XML* index and the *IR* index.

3.1 The taxonomy

We created a simple taxonomy, covering relatively wide area of possible questions. It's purpose is primarily to determine the semantic type of the expected answer (eg. location, duration, currency, name, ...) and secondary to provide an extendable natural language interface for retrieving data with already known semantic types (as tagged in the *XML* structure).

The database of patterns of the taxonomy is formed by several sets of regular expressions, each set corresponding to one possible answer type. The regular expressions from each set are sequentially applied to the input question until one of the patterns matches.

Example of patterns (adapted for English):

```
(when was)(.*)(born)
(did)(.*)(already die)
```

The '(.*)' construct should capture the actual entity, corresponding to the object of the question. This captured entity can then be used as a query to the *XML* index.

Each regular expression of the taxonomy can be optionally followed by the description (in a special query language) of the algorithm capable of retrieving the answer from the *XML* structure. The *XML* index module is bypassed if such an algorithm is missing.

Example:

```
(when was)(.*)(born)   MAN_name   @birth_date(MAN_born)
```

The final output from this module then looks like this:

```
<Q class="time" algorithm="MAN_name @birth_date(MAN_born)">When was <OBJECT>Jan Amos
Komenský</OBJECT> born</Q>
```

3.2 The XML index

The purpose of this module is to look up the object of the question in the corresponding index, to perform the retrieving algorithm and to provide an answer if nothing went wrong.

For example, if we consider the question "When was Jan Amos Komenský born?", the object of the question "Jan Amos Komenský" will be looked up in the contents of the '*name*' attributes of all <*MAN*> tags. The corresponding '*born*' tag (corresponding in the framework of the surrounding <*MEANING*> tag[1]) will then be retrieved and prettyprinted by the '*birth_date*' macro.

There are indexes such as *MAN_name*, *CITY_name* or *COUNTRY_capital* that provide mapping of every occurence of the corresponding entity (like 'the name of the MAN') to the

[1]the <*MEANING*> tag encloses different possible meanings of an encyclopedia entry

offset of it's enclosing encyclopedia entry in the *XML* text file. The first part of the name allways indicates the *XML* tag and the second part the indexed attribute.

All keys that we try for querying the hashes are derived from the object phrase as follows: Every word of the object phrase is normalized and converted to it's lemma. Then for every word-rotation of the phrase all it's possible word-prefixes are used to query the hash.

The indexes are currently implemented as internal Perl hashes, what implies excellent retrieval speed, but using the SQL database would be more memory-efficient and would provide greater query abilities and strength of our 'algorithm query language'.

3.3 The IR index

This module increases the overall robustness of the system. It is based on measuring similarity between words of the question and fragments of the encyclopedia, that could possibly contain the right answer.

We have implemented two different methods for building a query from the input question. The first is extremely simple — the whole question phrase is taken as a query and presence of every word in retrieved documents is *not* demanded. The most 50 similar documents[2] (in the terms of slightly modified inner product similarity function of the *CZIndex* system [4]) are retrieved. This method was, as expected, outperformed by the latter — feedback based, adapted from the *AT&T* engine [2].

The second method picks a few 'important' words from the question as the query and demands the presence of all those words in the documents. The query is then optimized via a feedback-based iterative algorithm. The feedback is controlled by the amount of documents that would meet the requirement of the given query. The query words are beeing added or removed on the basis of several heuristic selection rules. Query that returns at least one document and at most 50 (preferebly 30) documents is considered to be satisfactory.

Now we have not more than 50 documents retrieved, sorted according to the similarity function and we hope at least one of them to contain the answer. It has to be identified. We will suppose that the context required for identification of the answer is generaly short — certainly shorter than 30 kb (the size of the longest document). So the next step is to perform some kind of segmentation.

The first tested segmentation algorithm was to treat whole documents as segments and it didn't work very well. The second one is based on the concept of a sliding window and a similar one (though more complex) was applied in the *Lasso* engine [1]. It looks for small text windows with whole sentence boundaries (possibly overlapping) and assigns them a score according to their heuristically expressed relation to selected question words. Only the best 50 are further processed.

The last algorithm of the whole chain is the segments analysis. The task is either to identify an entity that probably represents the final answer or at least to point out segments with the greatest probability of containing the answer. The segments are parsed by the set of entity-recognition rules and all recognized entities are tagged. The rules are expressed in a similar way to the taxonomy — as regular expressions. They can be matched against either the plaintext variant of segments or the *XML* tagged variant of segments, where the *XML* tagging provides the morphology information of individual words (in the notation of the `ajka` analyser [5]).

[2]The single meanings of entries of the encyclopedia will be called 'documents' from now.

Example of the rules for the *duration* and *number* entities:

```
        <PLAIN>( )(in the years \d+ - \d+)()    duration
<MORPH>()((?:<W tags="[^"]*k4[^"]*">[^<]*</W>)+)()    number
```

The recognized entities are weighted and sorted according to these weights and the top 5 of them are presented as the final answer. The weights are computed as linear combinations of 7 different criteria (with heuristically determined coeficients) including the similarity scores of enclosing documents, the scores of enclosing segments and the frequencies of the question words occurences in the entity surroundings.

4 Evaluation

Here we present the performance of the system above the set of 123 test questions (generally complex ones and not simply answerable by the *XML* index module), while we used the second variants of both the query building and segmenting algorithms. The primary answer variant here indicates that the answer was short (entity-based). The answer is titled secondary if it was visually recognizable by the user in the returned segment of text. The answers were considered right if at least one of the top 5 returned answers contained the right information.

	Right answers
primary answer	31
secondary answer	71

Table 1: The evaluation, 123 test question

5 Conclusions

The results show that we can give precise and short answer in about 25 % of all cases and longer but satisfactory answer in about 58 % of all cases. This is not very stunning result, but reasonably good for the technology used. The performance relies on a number of heuristics, many of which were only basically set up and not very thoroughly fine tuned. There still remains big room for some optimisation work and even for inducing new technologies. The fail-case analysis shows the system could potentially benefit from utilizing the Czech WordNet lexikal database for e.g. term expansion, from improvements in the names handling of the morphology analysator, from word sense desambiguation and from full syntax analysis.

References

[1] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, and V. Rus, *Lasso: A tool for surfing the answer net*, In Proceedings of Text REtrieval Conference (TREC-8), 1999.

[2] A. Singhal, S. Abney, M. Bacchiani, M. Collins, D. Hindlea and F. Pereira, *AT&T at TREC-8*, In Proceedings of Text REtrieval Conference (TREC-8), 1999.

[3] Zdeněk Svoboda, *Znalec encyklopedie*, Master thesis, Faculty of Informatics, Masaryk University Brno, 2001. In Czech.

[4] Martin Povolný, *Začlenění jazykových nástrojů do systémů pro indexování dokumentů*, Master thesis, Faculty of Informatics, Masaryk University Brno, 2001. In Czech.

[5] Radek Sedláček, *Morfologický analyzátor češtiny*, Master thesis, Faculty of Informatics, Masaryk University Brno, 1999. In Czech.