Intelligent Information Systems 9999
ISBN 666-666-666, pages 1–11

Named Entity Recognition in a Polish Question Answering System

Marcin Walas and Krzysztof Jassem

Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland

Abstract

The paper discusses the impact of Named Entity Recognition (NER) on the results of a QA system with the Polish language. In the first part of the paper we describe a developing QA system, hipisek (www.hipisek.pl) that searches for the answer in the database of web-page articles. The system uses Information Retrieval techniques based on the keyword method. In the second part we describe how our self-developed NER tool, based on SPADE (Buczyński and Przepiórkowski, 2008), applied in the system. We report the results of the evaluation test, where we measure the impact of Named Entities Recognition on the overall performance of the system.

1 Introduction

Question Answering (QA) is a task related to Information Extraction (IE). QA systems accept queries expressed in a natural language and return a natural language answer, as precise and short as possible. Several methodologies for finding the answer in QA systems have been developed so far but according to (Feng, 2008) a typical model of a QA system is composed of the following elements:

1. Question processing
2. Question classification
3. Question reformulation
4. Document retrieval
5. Question extraction

QA systems consult some kind of knowledge database, which may be represented in one of the forms:

- Relational database (Sneiders, 2002);
- Knowledge ontology (Atzeni *et al.*, 2004), (Cowie *et al.*, 2000); such systems are called ontology based QA systems;
- Internet (Katz *et al.*, 2003), (Zheng, 2002) – called web-based QA systems.

Depending on the depth of semantic analysis and complexity of knowledge database we may distinguish between:

- Shallow QA systems;
- Deep QA systems.

Shallow QA systems use standard Information Retrieval methods. Brill et al (Brill *et al.*, 2002) introduce a method that sends a set of queries to an Internet search engine and then statistically processes the set of returned documents. Another approach, used for answering encyclopedic questions, applies answer templates, which are generated automatically (Lee *et al.*, 2005), (Ravichandran and Hovy, 2002). The AnswerBus system (Zheng, 2002) returns a short fragment of a text, which contains the answer.

Deep QA systems carry out more refined question analysis and processing. The goals is to maximise the answer precision and correctness. Such systems apply methods for semantic analysis. Shari and Li introduce the use of Named Entity Recognition module in a QA system (Srihari and Li, 1999). Named Entities play the crucial role in extracting the answer also in the system described in (Noguera *et al.*, 2005).

There exist very few QA systems dealing with the Polish language (only one on-line Polish QA service is known to the authors, namely www.ktoco.pl). The aim of our project is to create a web-based Polish QA system that carries out some semantic processing. We have implemented a shallow method based on the Internet query formulation. We have enriched our system with deep elements, such as use of several thesauri and a tool for Named Entity Recognition. We have limited ourselves to questions about time, space and person. Section 2. is a general overview of the system. Section 3. provides some details on question processing. Section 4. deals with the process of extracting answers. Section 5. reports how the NER module is implemented in the system. Section 6. gives evaluation of the impact of the NER module on the performance of the system. We conclude with the reference to future plans.

2 System overview

Our project is an emerging Polish QA system (www.hipisek.pl), which processes questions asked in the Polish language and searches for the answer in the knowledge database acquired automatically from plain-text documents concerning press news.

In our system we use linguistic tools and resources in the process of question analysis. We use Information Retrieval algorithms in the search for the answer. We assume that the answer is a sentence (or a set of sentences) occurring in the indexed knowledge database – we do not attempt to reformulate the answer. Our approach consists in transforming the question to a set of search engine queries, the response to which forms a small set of hypothetical answers, which are then ranked for their assessed relevancy to the question.

Knowledge database is created by means of a web-crawler program, which automatically extracts plain-text documents excerpted from the articles on the Polish news websites. Each document in our database is equipped with the following metainformation:

- Title – the title of the article;

- Stamp – the date of publishing of the article;
- Keywords – keywords attached to the article by the website;
- Url – url of the article.

Metainformation is acquired by our web-crawler directly from the indexed webpage of the article. Articles are indexed with the free search engine indexing machine Sphinx (Aksyonoff, 2001). The system processes the following types of questions:

- Who-questions (e.g. *Who is the Prime Minister of Russia?*);
- Questions about time (e.g. *When were the first Olympic Games held?*);
- Questions about place of the event (e.g.. *Where were the first Olympic Games held?*).

The question is processed in the following process:

1. Question is translated into our self-developed formalism QQuery;
2. QQuery is translated into a set of search-engine queries;
3. Using the Sphinx search-engine a set of documents is extracted from our knowledge base, (candidate document set);
4. From the candidate document set the system extracts sentence parts, which are most likely to contain the answer (candidate answer set);
5. All sentences from the candidate answer set are scored and the best scored sentences (together with their two neighbouring sentences – the proceeding and succeeding sentence) are returned as the answers to the processed question (answers scoring is outlined in Subsection 4.2).

In our approach we extend the keyword based question processing method presented in (Zheng, 2002) and (Harabagiu and Bejan, 2005).

3 Question Processing

In the first step of the processing we identify base forms of all words in the sentence. We use our own automatically acquired Polish words lexicon *Dylemat* (Graliński and Walas, 2009) in the process.

In next step we classify each word into one of the four classes:

- topic (a subject of the question);
- action (a main verb in the question);
- important lexemes (containing useful information in further question processing);
- redundant lexemes (i.e. prepositions).

Formally speaking, we translate the question into a structure (called QQuery) containing the following entities:

- `TOPIC` – the subject of question, which should be repeated in the answer;

- `ACTION` – the main activity of the `TOPIC`;
- `CONSTRAINTS` – the rest of the lexemes from the question, containing important information (excluding redundant lexemes);
- `TYPE` – type of the question.

For example the question: *Gdzie wczoraj prezydent Lech Kaczyński spotkał się z ministrem sprawiedliwości?* (eng. *Where did president Lech Kaczyński meet the Minister of Justice yesterday?*) is parsed to following structure:

```
Type: PLACE PAST
Topic: prezydent[prezydent] Lech[Lech] Kaczyński[Kaczyński]
Action: spotkać[spotkał]
Constraints:
   minister[ministrem]
   sprawiedliwość[sprawiedliwości]
```

We define three question types: `TIME` (questions about time), `PLACE` (question about the place of the event) and `PERSON` (questions about people). Moreover, we add a subtype of the question, which refers to the time context of the question: `PRESENT` (current information), `PAST` (historical or out-of-date information). In some cases the question type remains undefined (when the question type is not supported in our system).

To perform the translation into the QQuery we first perform linguistic analysis of the given question. Once all base forms of sentence tokens have been identified, we search for their synonyms (in our thesauri) in order to later record them in the QQuery structure.

Two methods are used in the translation:

- Rule-based;
- Heuristics-based

3.1 Rule-based translation

Each rule consists of the match clause and the translation clause. The formalism for rule description is based on SPADE (Buczyński and Przepiórkowski, 2008). The match clause is a sort of a question template consisting of a number of match conditions, which all have to be fulfilled by the analyzed question. For example, the following match clause accepts questions of the type: *Gdzie teraz jest ktoś?* (eng. *Where is somebody now?*):

```
Match: <normal=gdzie> <normal=teraz> <base=być>
       <sem~name> <used~^{UPPER_CASE}>?
```

A question matches the above rule if the first token can be normalized (by downcasing) to *gdzie* (eng. *where*), the second token can be normalized to *teraz* (eng. *now*), the third is a lexeme with the base form *być* (eng. *to be*), the fourth token should be a person name, and the last token is optional beginning with the upper-case letter (possibly a surname).

The translation clause is an instruction on how to determine the question type and correspondence between tokens and the QQuery properties (e.g. token *teraz* determines the question type as `PRESENT`).

The next step is translation from the QQuery into the set of search-engine queries, where we use the reformulation method, introduced in (Tomuro and Tomuro, 2003). Hovewer, we do not perform full question reformulation. Instead, we automatically create small fragments of the reformulated question that are likely to occur near the answer. These are called *search phrases*. Search phrases are written to another QQuery entity: `PHRASES` entity.

3.2 Heuristic-based translation

If processed question doesn't match any of defined rules, then heuristic-based translation method is used.

The method assigns scores to all lexemes from the input questions, which estimate their likelihood of belonging to an appropriate entity of the QQuery. The scores take into account: part of speech, position in the sentence and provided semantic information.

In particular we assumed that:

- Question topic is most likely a noun;
- Question action is most likely a verb;
- If named entity referring to object (i.e. place, person) occurs in question, it is likely that it is a question topic;
- If lexeme with semantical information referring to object (i.e. place, person) occurs in question, it is likely that it is a question topic;
- Question topic and action occur often near the beginning of the sentence (this is based on our observations of random parts of questions corpus collected by our system in development phase).

We assigned each factor a weight from a set of $[0, 1]$ (basing on experiments). For each word we compute a score for being a topic or action of the question. Two best scored words are chosen.

4 Extracting Answers

We define an answer as a sentence containing information satisfying user information need. Moreover we provide an answer with the context defined as two neighbouring sentences (proceeding and succeeding).

Extraction of answers is divided into two phases:

- Retrieving documents, which are likely to contain answers (candidate document set);
- Extracting sentences, which are most likely to contain the answer, from the candidate document set.

4.1 Document retrieval

The goal is to identify a possibly small set of documents that should contain the answer. We make the following assumptions:

- The neighbourhood of the answer contains one of the search phrases;
- The neighbourhood of the answer contains `TOPIC` and `ACTION`;
- The neighbourhood of the answer *"is full"* of lexemes occurring in the question or their synonyms.

Basing on the above assumptions we generate the following three types of search-engine queries:

- Phrase-clause-based query;
- Topic-based query;
- Bag-of-words query.

The phrase-clause-query is composed as an disjunction of all phrases from the `PHRASES` entity.

The topic-based query is a conjunction of the following elements:

- phrases of the form: `TOPIC-ACTION`;
- alternative of `TOPIC` and its synonyms;
- selected lexemes from the `CONSTRAINTS` entity;

The bag-of-words query is simply a concatenation of all lexemes in `TOPIC`, `ACTION` and `CONSTRAINTS` as well as their synonyms. The document candidate must contain `TOPIC` and at least half of the remaining lexemes.

The found documents are ranked with respect to scores returned by Sphinx. Sphinx scores are computed with the use of Okapi BM25 weighting scheme. Okapi BM25 takes into account frequency of term from query in whole indexed document collection (i.e. if document contains less frequent terms its rank is higher). For details see (Manning *et al.*, 2008).

Ten highest ranked documents are forwarded to further processing (see Section 4.2).

4.2 Answer extraction

Each candidate sentence is scored with two marks: acceptance score and quality score. The acceptance score must exceed a given threshold value (fixed for each question type), to qualify a document as a potential answer. The quality score is used to rank potential answers in the order of their relevance to the question.

The following factors are taken into account in the scores:

- Method of sentence retrieval;
- Occurrence of the question topic (or its synonym);
- Occurrence of the question action (or its synonym);
- Number of question constraints satisfied by the answer (a question constraint is satisfied by the answer if constraint's lexeme or its synonyme occurs in the answer);

- Quality of the source document (which is a fixed weight assigned to source website of the document);
- Score of the sentence's neighbourhood (one sentence before and one after);
- Occurrence of question topic in the keywords and topic of the source document;
- Publishing date.

Each factor is measured with the score from the set $[0; 1]$. The scores are aggregated using the weighted sum.

5 Named Entity Recognition in QA

We believe that deeper semantic analysis of the question can increase the precision of the answers. We have chosen to apply a NER module as the first step of our system upgrade. Named Entities may be defined as continuous parts of a text containing semantic information (i.e. address, organization names, date), and we expect from a NER module to mark Named Entities boundaries in the text and classify their types.

We have developed a rule-based NER tool for the Polish language on the basis of the NERT environment described in (Graliński *et al.*, 2009). The rules include semantic conditions that may be verified by free sources available online such as: TERYT (national register of administration division), list of Polish names, DBPedia (a database of structured information acquired from Wikipedia Project).

We assume that answers for specific types of questions should contain specific types of Named Entities, for example:

- Who-question's answer should contain a `PERSON` named entity;
- Where-question's answer should contain a `PLACE` named entity;
- When-question's answer should contain a `TIME` named entity.

We have applied these restrictions to our answer extraction module. Namely, we have added an additional acceptance rule saying that the answer (or its neighbourhood) for the specific type of the question must contain the corresponding Named Entity.

6 Evaluation

We have performed the following evaluation experiments:

- Usability test;
- Expected information test;

In all tests we restricted the test to types of questions that are processed by our QA system. Moreover, we restricted the domain to facts included in our knowledge database.

6.1 Usability test

The usability test aimed at comparison between our QA system, Polish open-domain QA system www.ktoco.pl and Google search engine treated as a QA system. The experiment applied user-centred evaluation (Ong *et al.*, 2008).

We have prepared a set of 25 test questions concerning information that may be found in sources indexed by our system. The answers of the three systems were ranked by five students. Each student send each of the questions from test set to each of the compared systems. Then each student ranked returned answers in the scale from 1 (worst) to 5 (best). If no answer was delivered by the system (or the system crashed) the answer was given zero points. The results of the test are shown in Table 1.

6.2 Expected information test

Expected information test is a kind of regression test based on the reference set of questions and information expected in the answer, following the idea of the TREC conference (Dang *et al.*, 2007). Expected information were coded as a set of regular expressions. Each regular expression was given a weight from the set between 0 and 1. The answer is scored with maximum weight of the set of regular expressions matching the answer's sentence. If more than one answer was given, all were considered (with a penalty factor of $\frac{1}{k}$ were k is the order of the given answer). If no regular expression was matched, the answer was marked as wrong (with the score of zero). If no answer was given, the answer was marked as `NULL` (with no score given).

The test set had to be prepared manually. We created the test set in the following way. We chose a set of 500 random sentences from the indexed corpus, with two neighbouring sentences. Each chunk was checked manually. If it contained essential information, we created a question for the chunk and a set of regular expressions for expected information. Finally we created a set of 65 questions with the expected information. A sample of the set is shown below:

```
Question: Gdzie siedzi Madoff?
1     qr{więzieni.*\s+.*Północn.*Karolin} # North Carolina Prison
0.2   qr{wiezieni}                        # prison
```

In our expected information test the questions were processed by our QA system (with NER module switched on and off) and ktoco.pl QA system. In test we

Table 1: Summarized results of Usability test

	Hipisek.pl	Ktoco.pl	Google
Total score	258	189	221
Average score	2,06	1,51	1,77
No answer	43	59	43
Perfect answer (mark 5)	20	14	13

computed:

- Answered questions – a number of questions with any answer given;
- Questions with correct answer – a number of questions with correct answer provided among all given;
- Answers provided – a total number of answers provided for all questions (both systems provide multiple answers for single question);
- Correct answers – a total number of correct answers among all given.

We used the F-score measure for system comparison, where precision and recall were calculated as follows:

$$\text{Precision} = \frac{\text{total score}}{\text{maximum possible score}}$$

$$\text{Recall} = \frac{\text{number of questions with any correct answer}}{\text{total number of questions}}$$

Results of our test are given in table 2

6.3 Comments

In both tests we achieved a better overall score of our system, than reference systems. However absolute scores are not satisfying. A slight diffrence between the usability test's score of Google and our system indicates that on the current stage of system development increase of usability is practically unnoticable (scores are better on average of 0,3 point).

The results of the second evaluation experiment show that our approach of question processing and extraction is reasonable. By adding NER module we could considerably increase the score obtained by our system (by approximately 40%).

Table 2: Results of expected information test

	Hipisek.pl NER	Hipisek.pl (no NER)	Ktoco.pl
Answered questions	44	49	64
Questions with correct answer	24	19	19
Answers provided	142	159	195
Correct answers	41	31	43
Precision	0,24	0,16	0,18
Recall	0,37	0,29	0,29
F-score	0,29	0,21	0,22

Detailed scores show that the main diffrence between ktoco system and our system is the number of answers returned. Ktoco system returns as many answers as possible which are unlikely correct. In our approach we provide less answers, with significant higher ratio of correct ones. Our system fulfills the general idea of QA systems of providing answers as precise as posible.

7 Conclusions and future plans

The paper reports the status of a developing QA system, hipisek. At the moment the system uses shallow QA methodology but our aim is to systematically incorporate deeper mechanisms. The first step was the usage of a NER tool. Evaluation tests imply that the effort significantly improved the quality of the answers. In the nearest future we intend to verify the influence of deep syntactic parsing on the performance of the system

References

Andrew Aksyonoff (2001), Sphinx - Free open-source SQL full-text search engine, www.sphinxsearch.com.

P. Atzeni, R. Basili, D. Hansen, P. Missier, P. Paggio, M. Pazienza, and F. Zanzotto (2004), Ontology-based Question Answering in a Federation of University Sites: the MOSES Case Study, in *Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems (NLDB)*, Manchester, United Kingdom.

Eric Brill, Susan Dumais, and Michele Banko (2002), An Analysis of the AskMSR Question-Answering System, in *Proceedings of EMNLP 2002*.

Aleksander Buczyński and Adam Przepiórkowski (2008), ♠ Demo: An Open Source Tool for Partial Parsing and Morphosyntactic Disambiguation, in *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, European Language Resources Association (ELRA), Marrakech, Morocco.

Jim Cowie, Evgeny Ludovik, Hugo Molina-Salgado, Sergei Nirenburg, and Svetlana Scheremetyeva (2000), Automatic Question Answering, Proceedings of the RIAO Conference.

Hoa Trang Dang, Jimmy Lin, and Diane Kelly (2007), Overview of the TREC 2007 Question Answering Track, in *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, Gaithersburg, Maryland.

Junlan Feng (2008), Question Answering with Question Answer Pairs on the Web.

Filip Graliński, Krzysztof Jassem, and Michał Marcińczuk (2009), An Environment for Named Entity Recognition and Translation, in *Proceedings of the 13th Annual Conference of the EAMT*, pp. 88–96, Barcelona.

Filip Graliński and Marcin Walas (2009), Looking for new words out there, in *Proceedings of the International Multiconference on Computer Science and Information Technology*, volume 4, pp. 213–218, Mrągowo.

A Harabagiu and Cosmin Adrian Bejan (2005), Question Answering Based on Temporal Inference, in *In Proceedings of the AAAI-2005 Workshop on Inference for Textual Question Answering.*

Boris Katz, Jimmy Lin, Daniel Loreto, Wesley Hildebr, Matthew Bilotti, Sue Felshin, Aaron Fern, Gregory Marton, and Federico Mora (2003), Integrating Web-based and corpus-based techniques for question answering, in *In Proceedings of the Twelfth Text REtrieval Conference (TREC*, pp. 426–435.

Changki Lee, Ji-Hyun Wang, Hyeon-Jin Kim, and Myung-Gil Jang (2005), Extracting Template for Knowledge-based Question-Answering Using Conditional Random Fields, in *Proc. 28th Annual International ACM SIGIR Workshop on MFIR*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze (2008), *Introduction to Information Retrieval*, Cambridge University Press.

Elisa Noguera, Antonio Toral, Fernando Llopis, and Rafael Muñoz (2005), Reducing Question Answering Input Data Using Named Entity Recognition, in *Proceedings of the 8th International Conference on Text, Speech & Dialogue*, pp. 428–434.

Chorng-Shyong Ong, Min-Yuh Day, Kuo-Tay Chen, and Wen-Lian Hsu (2008), User-centered evaluation of question answering systems, in *ISI*, pp. 286–287.

Deepak Ravichandran and Eduard Hovy (2002), Learning Surface Text Patterns for a Question Answering System.

Eriks Sneiders (2002), *Automated Question Answering Using Question Templates that Cover the Conceptual Model of the Database*, Lecture Notes in Computer Science, Springer Berlin/Heidelberg.

Rohini Srihari and Wei Li (1999), Information Extraction Supported Question Answering, in *In Proceedings of the Eighth Text REtrieval Conference (TREC-8*, pp. 185–196.

Paraphrases Noriko Tomuro and Noriko Tomuro (2003), Interrogative Reformulation Patterns and Acquisition of Question, in *In Proceedings of IWP 2003*.

Zhiping Zheng (2002), AnswerBus Question Answering System.