Artificial Intelligence 9999
ISBN 666-666-666, pages 1–18

Issues of Polish Question Answering

Piotr Przybyła

Institute of Computer Science, Polish Academy of Sciences,
ul. Jana Kazimierza 5, 01-248 Warszawa, Poland,
P.Przybyla@phd.ipipan.waw.pl

Abstract

In this paper, perspectives for building a Polish question answering (QA) system are considered. It begins with an explanation of an increasing demand for such systems, along with inevitable difficulties and usage examples. Then, a question answering task is defined as a point of reference and for future evaluation. Existing solutions of similar tasks for English, evaluated during text processing conferences, are presented along with their common architectural scheme. To assess the applicability of those solutions to Polish, their language dependence is discussed, preceded by an outline of distinctive features of Slavonic languages, especially Polish, and their impact on text processing. Some already existing QA solutions for Polish are also enumerated.

Keywords: question answering, natural language processing, Polish, information extraction

1 Introduction

Question Answering (QA) is a challenge present in computer science since its beginnings. It is a natural human desire: to have a purely automatic tool that will help in solving problems with information just like mechanical tools solve problems with matter. The invention of first computers sparked off numerous visions of *talking machines*, playing important roles in popular culture. It seemed to be the most obvious direction of development for *electronic brains*. From that point of view, the following definition could be used:

Definition 1 ***Question Answering System*** *is a computer system, capable of addressing questions formulated by a human user in his natural language.*

Why this feature of computer systems is so important for users and so neglected by their designers? To understand it, one needs to realise that computers think in their own way, and the burden of translating the problem to a computer-understandable way usually lies on user's shoulders. The poorer are the computer skills of the user, the harder it becomes.

For example consider the task of searching the WWW in order to find a specific piece of information. This service is a basic tool for Internet users, including the least experienced ones. However, designing a successful query involves obeying certain rules. Some of them are counterintuitive (e.g. choosing words which will

not appear in web pages that will *not* contain desired information), some involve additional search tools. Unfortunately, most users are not aware of them, which leads to the following results:

- every day, 3 million queries to the *Yandex* (a Russian search engine) are plain questions [1] (*How to lose weight?* is the most popular),
- 80% of the users do not use boolean operators at all [2],
- other advanced features of search engines (wildcards, etc.) are also seldom used.

This type of problems are encountered whenever knowledge stored in a computer system needs to be accessed by a person, who is not qualified enough (although he may be an expert in his own field), e.g. in medical systems, customer services or libraries.

In fact, almost all applications could benefit from adding question answering elements. Even regarding those with low level of interaction with user, such as weather modelling or engineering calculations, an underlying problem could be formulated as simple questions in a natural language, like *Will it rain tomorrow?* and *Is this bridge able to hold the expected load?*, respectively. More about such modules could be found in [3].

1.1 Difficulties

Now, let us focus on another side of the problem: why so desirable feature is so rarely implemented in commercial products? That is because its realisation is very hard; Shapiro describes it as AI-complete. It means that *solving the problem of the area is equivalent to solving the entire AI problem - producing a generally intelligent computer program* [4]. Features that make the natural language processing (especially question answering) so challenging are the following:

- **contextuality**: The meaning of every language element (a word, a sentence) depends highly on its neighbourhood. It makes Information Retrieval (IR) and Information Extraction (IE) hard, as information obtained by a user may be different than expected because of its context. Usually, in those cases, systems provide the user with additional information about the context, such as text snippets in web search engines. The most problematic contextual elements i.e. ellipses[1] and anaphoric expressions[2] may be dealt with in QA systems by specialized modules, called anaphora resolution solutions [5].
- **ambiguity**: Sometimes even while knowing a context of a particular sentence, we still have a number of possible ways to understand it. It could be unintended, but also may be a deliberate act of an author in order to influence a recipient, e.g. by suggesting instead of stating explicitly. Usually, it is assumed that such situations should not appear in well-formed, informative texts.

[1] Ellipsis is an omission of a word (or more) in a text, which could be easily determined from a context.

[2] Anaphora is an expression with a certain word (or a whole group of words) replaced by another word, usually a pronoun.

- **imprecision**: It seems obvious that human utterances lack the precision of, say, mathematical formulae or numerical computations. However, it is not a problem as long as questions which are to be answered using this knowledge are of similar precision. Again, choosing a textual information source adequate to our needs is crucial.
- **implicitness**: In communication between humans, a message being transferred is not stated explicitly, but rather derived from all the knowledge common for its sender and receiver, which words are only a part of. Other relevant knowledge sources include gesticulation, facial expression and tone of voice. Unfortunately, even avoiding the above by working with digital texts, we still have to take into account another hidden source of knowledge: a common sense. For example, let us consider the following pair of sentences: *Elephants ate all the fruits yesterday. They seem very content with it.* To understand it, we need to resolve the anaphora *they* in the second sentence. Unfortunately, it is not possible on syntactic level; *elephants* and *fruits* are both acceptable candidates in this situation. Obviously, any human reader knows that he should choose the *elephants*, as the *fruits* are unable to have any feelings, including the contentness. However, this information is not provided by the context; it's also highly unlikely to appear in any conventional textual source, such as a newspaper archive or an encyclopaedia. It is expected to be a part of world understanding, usually called commonsense knowledge, including which in a computer system poses a great challenge.

The problems of contextuality and implicitness in a QA task are easier to deal with when using appropriate linguistic tools; some of them are discussed in section 4.4. A careful choice of source texts for QA helps to overcome difficulties with ambiguity and imprecision, see section 2.2.

1.2 Paper aim and organization

In section 2 a clear, language-independent problem definition is presented to focus on core problems, namely the natural language processing (NLP). Next, in section 3, solutions and approaches present in the literature are enumerated including a common architectural scheme with detailed explanation of its elements. Section 4 is devoted to discussing the relevancy of previous statements to the Slavonic languages, especially Polish. In particular, a few existing QA solutions are presented and the architectural scheme from section 3.3 is analysed to identify language-dependent elements and discuss their availability in Polish. Section 5 concludes the paper.

2 Problem definition

In the previous section, the motivation for building QA systems was outlined — a huge number of computer systems could improve their users' satisfaction by implementing natural language communication. The purpose of this section is to define a QA problem in order to focus our attention on its most important features and set aside problems related with a particular use. Generally, the task which is to

be defined here is called an **open domain factoid question answering using text corpus**. Let us state the following:

Definition 2 ***Question Answering*** *is a problem of automatic extraction of a concise answer in a* ***natural language****, in response to a question formulated in the* ***natural language****, based on knowledge gathered from collection of texts in the* ***natural language****.*

It is a much narrower formulation than in the Definition 1; in this case not only the questions, but also the answers and the sources are expressed in a natural language. However, as it includes the core language processing problem, its solutions could also be used in implementations of similar tasks, say, interface of a database.

2.1 Questions

Let us define a Question in our QA system as a sentence, being a question according to natural language rules, expected answer to which is a simple entity, i.e.:

- An answer to the question is present in the text collection, but it may be indirect, or preparing it may involve combining knowledge from several documents.
- Questions should not require any complex reasoning or computing, so the focus remains on text processing. For example the following question: *Which is the largest country of those that spend more than 10% of their budget on their army?* would not be accepted, unless there is a corresponding sentence in a source text.
- A target of question is a simple entity, regardless whether named (person, city, country, date, quantity, etc.) or unnamed (thing, concept, property, animal, event, action etc.). Such question are usually called **factoid questions**. Queries demanding a definition or an explanation, such as *What is a global warming?* do not fall into this category.
- The queries are grammatically correct questions in the natural language, which rules out requests like *Tell me the name of the first post-war president of France.*

A test set for a QA task is just a set of questions as defined above; no additional information, e.g. about context or domain, is provided, thus the questions need to be self-explaining and precise.

2.2 Texts

As stated in section 1.1, a level of difficulty of a QA task depends highly on texts to be processed. An ideal textual database for evaluating such systems would have the following properties:

- **linguistic correctness**: Usually texts that people describe as well-written and correct are also relatively easy for automatic processing.
- **precision**: To extract valuable information from the texts, we need them to be written precisely, without ambiguities or vague statements.

- **informativeness**: To be able to answer interesting, non-trivial and worth asking questions, we need the corresponding source to contain appropriate answers. That rules out all sort of literary work, albeit satisfying the remaining conditions.
- **abundance**: Modern NLP challenges are mostly connected with a need to deal with huge numbers of texts — to evaluate a QA system we want it to process a significant knowledge base, containing an overwhelming amount of information, mostly irrelevant to any query.
- **open-domain**: Closed-domain systems are being actively developed, but employ different techniques (such as manual creation of possible questions list or a knowledge base). Herein, only open-domain systems are considered, which needs to be reflected in the knowledge source.

Sources which are used for similar purposes in literature are newswire bases or encyclopaedias. They come from relatively reliable sources and satisfy all the conditions outlined above. On the other hand, they are very far from what may be found in the most important "battleground" of the modern text processing: the Internet. The content of web pages is usually imprecise, ambiguous, more persuasive or emotional than informative, often incorrect in terms of syntax or spelling. The *Wikipedia* may be some kind of a compromise in that matter: offers highly-informative articles, but written by internet users instead of professional journalists or scientists.

2.3 Answers

What exactly is an answer to our question? For example, let us consider the question *When did Albert Einstein marry his second wife?*. In fact, the four answer levels are possible:[3]

1. A document containing an answer: *Wikipedia:Albert_Einstein*. However, such an answer is rather typical for Information Retrieval systems, as it still requires a user to extract the entity from the text.
2. An answering sentence: *Einstein married Elsa Löwenthal (née Einstein) on 2 June 1919, after having had a relationship with her since 1912.* Here, there is less work for a user, but he still needs to find the dates in the sentence and decide, which corresponds to his question.
3. Only an entity demanded: *2 June 1919*. That seems to be the option most convenient for a user.
4. A full phrase based on a question: *Albert Einstein married his second wife on 2 June 1919.* Although that would be a natural way of responding, it is seldom implemented in existing open-domain QA systems.

Henceforth, in this paper by *answer* the entity demanded, as in the point 3, will be meant.

[3]Answers extracted from: `http://en.wikipedia.org/wiki/Albert_Einstein`.

2.4 Evaluation

Generally, an evaluation of a QA system is a process of measuring its outcome in a precisely defined environment. The environment consists of questions, sources and answers satisfying certain conditions, for example those outlined in sections 2.1, 2.2 and 2.3 of this document. What is more, one needs to define measures to be applied to gathered answers. Aside from obvious computing a percentage of questions answered correctly, worth mentioning are number of questions a tested system decided to answer (crucial in *Jeopardy!* competition, in which QA system named *Watson* took part [6]) and adequacy of a document returned as a support of an answer (e.g. TREC competition [7]).

3 Related work

The field of question answering attracts a lot of attention, resulting in many interesting solutions. However, while their authors use different testing environments or different correctness measures, they remain incomparable. One of the ways to overcome this problem are open competitions organised together with corresponding conferences.

3.1 Open competitions

The most important competition was *Question Answering track* at *Text REtrieval Conference (TREC)*, which was arranged in years 1999-2007 [7]. In 2007, a total of 515 questions was organised in 70 series, each related to a certain target (e.g. *Guiness Brewery*, *Jon Bon Jovi* or *March Madness 2011*). The following question types were present[4]:

- **FACTOID**: an expected result type is a single entity, being an answer to the question, e.g. *On what date did this earthquake strike?*,
- **LIST**: an expected result type is a list of entities satisfying the question, e.g. *What countries were affected by this earthquake?*,
- **OTHER**: an expected answer type is an interesting information *nugget* about the target, different from those present in the remaining questions from the series.

In 2008, the competition transformed into *Opinion Question Answering* at *First Text Analysis Conference (TAC 2008)*, where the task was to find opinions in a blog corpus [8]. The *Question Answering for Machine Reading Evaluation* has been offered at *Conference and Labs of the Evaluation Forum*. In this case answers were given (5 options for each of 160 questions), so the focus was on a deep text understanding, namely not only finding entities in text, but also reasoning and inferring based on extracted knowledge [9]. Tasks (i.e. source texts, questions and answers) were given in English, German, Italian, Romanian, Spanish, Arabic and Bulgarian. Another multilingual task was the *Cross-Lingual Question Answering (CLQA) Task* at *NTCIR (NII Test Collection for IR Systems) Workshop*. An aim

[4]Examples from the target *Pakistan earthquakes of October 2005*, published in [7].

was to answer questions with named entities extracted from corpora in Chinese, Japanese and English [10].

Another idea worth mentioning has been proposed by a team working on *Watson* [6] at IBM. In [11] they proposed to build so-called *Open Collaboration Framework* not only in order to compare results, but also to invigorate collaboration between academic and industrial organisations by using shared system model and open-source components and exchanging elements of QA solutions.

3.2 Solutions

The first solutions for problems formulated in a way similar to section 2 were proposed in the 60's (a detailed review may be found in [12]). However, they relied mostly on transforming a text and questions into some kind of a formal representation and then employing rapidly developing theorem-proving techniques to find an answer. Unfortunately, this approach did not lead to satisfying results not only because of the ambiguity and imprecision of English, but also because of a lack of reliable linguistic tools necessary for such a transformation. In this situation, researchers needed to base on few manually-prepared rules covering only a small part of the language. The beginnings of the Internet encouraged researchers to use similar approach again, this time to extract information from the WWW [13].

A large number of modern QA systems took part in the TREC QA competition in years 1999-2007 (overviews published in [14, 15, 16, 17, 18, 19, 20, 21, 7]). Most of participating solutions had a similar architecture, outlined in the next section, and used available linguistic tools and resources heavily.

The last breakthrough worth mentioning was made by a team from the IBM Research Division, who managed to build *Watson*, a complex QA system, which participated live in the U.S. TV quiz show called *Jeopardy!* [6]. The task differs from what is described above in three ways: first, queries are not formulated as pure questions, but *clues*, to which a participant is expected to respond, asking an appropriate question. Secondly, usually it is not enough to have pure factoid knowledge; one needs to combine distant relations, solve puzzles, understand word games. Finally, during competition, each of the participants needs to decide whether he wants to respond to a clue after seeing it; therefore, a precise confidence estimation becomes crucial.

3.3 Common architecture

As stated above, a majority of modern QA solutions share a common architectural scheme, which has been illustrated in Figure 1.

As we can see in the figure, a whole process starts with a pre-processing of a text base, which is to be executed offline, before the system is presented to users. When a question is available, it undergoes a similar processing stage, which also determines its type. The information is used in subsequent steps: searching for relevant documents using an index, choosing a sentence which matches the question best, and, finally, extracting a demanded entity from the sentence.

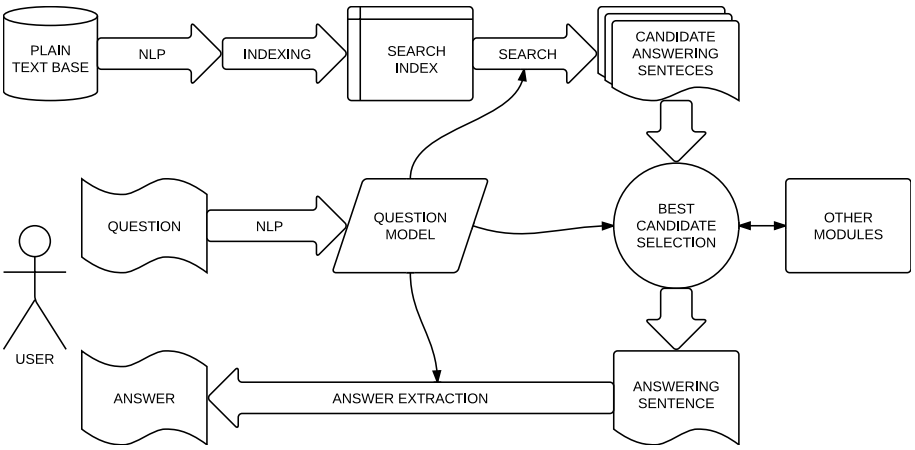

FIGURE 1: Outline of a typical QA system.

3.3.1 Text base processing

Before an evaluation begins, the text corpus is processed to facilitate its usage as a knowledge base for question answering. For English sources, usually a *stemming*[5] is applied, but in case of languages with the nominal inflection it may get more complicated (see section 4.1). Systems employing a deep semantic parsing may also execute it as a stage of the text preprocessing (e.g. [22]).

3.3.2 Index

An index is introduced because of performance reasons. Regardless of a sentence selection method, it is not possible to apply it to all of the documents from the corpus. Therefore, in the preparatory stage, the full-text search index is created, using one of powerful and freely available search engines, such as *Apache Lucene*. When the question is processed, it is transformed into a list of keywords, forming a search query. This transformation process usually includes a removal of stop-words[6] and using external resources for query expansion by adding synonyms [23] or more complex semantic transformations [22]. A certain number of top results is then passed on to the subsequent steps. Some of the systems, such as [24], include feedback loops, returning to this stage in case of extracted documents being irrelevant.

3.3.3 Question processing

There are three goals of this stage:

[5] *Stemming* is a process of assigning a pseudo-word (called a *stem*) to words in a text in a way that will guarantee (or, more likely, make very probable) that different forms of the same word will get equal stems.

[6] The term *stop-words* refers to words, which are very frequent in all documents in a particular language, such as (for English) *a*, *the* or *is*.

1. question type determination,
2. answer type determination,
3. transformation to a search query.

The first task is to find out what is a general type of the question: factoid, list, definition, yes-no or other. In the TREC QA task this information was being given explicitly [7], but it is not the case in real-world applications. The second stage is an interesting problem itself: how to determine what is the type of an entity, which is sought by the questioner? The interrogative pronouns (e.g. *who*, *where*, *when*) may be of some use (*who* usually refers to a person), but they are not enough (*who* question may also be answered by an entity being an organisation). Usually solutions include a number of possible target categories (e.g. PERSON, DATE, TITLE, QUANTITY, CITY, ...) or even complex hierarchies, like in [25]. Apart from rules created manually some systems use the WordNet[7] [24, 27] or machine learning techniques [25, 28] to find the answer type. Requiring the user to fill in a form instead of writing the question, like in [29], renders question processing unnecessary.

3.3.4 Sentence selection

The core element of every QA system is responsible for choosing a sentence which is most likely to contain the answer. Four main approaches to the problem may be identified:

1. Pattern matching,
2. Semantic matching,
3. Probabilistic matching,
4. Conversion into logic forms.

Pattern matching is an extremely simple and surprisingly effective technique for question answering which has been used extensively. For example, let us consider the following question: *When did Frederick I Barbarossa die?*. It is quite probable that the answering sentence will take the form: *Frederick I Barbarossa died on <DATE>*. We can then scan the sources very efficiently, just looking for the specified pattern. Of course, if this information is expressed differently, then the approach fails. However, if we use the WWW as a corpus, then we may exploit its redundancy; the same information appearing in different web pages, expressed in different ways, increases the probability of finding our pattern. Apart from creating those patterns by hand, we may also automatically acquire them from the Web [30, 31]. This approach has also been successively employed when focusing on a particular question type, namely definition questions [32].

The use of the pattern matching is always limited by a variety of the natural languages; that is why semantic matching is being employed nowadays. In this approach we still match the question to a candidate sentence, but the connection between (potentially) corresponding words is no longer a pure equality, but can be

[7] *Princeton WordNet*[26] is a lexical database containing words organized in synonymous sets and connected by semantic relations, such as hypernymy, hyponymy or meronymy (part-whole relation).

derived from a lexical database like the WordNet. For example, while answering the question mentioned previously (*When did Frederick I Barbarossa die?*) we should also accept *On <DATE> Frederick I Barbarossa passed away* as an answer because *to pass away* and *to die* are synonyms, but also *Frederick I Barbarossa drowned in the river on <DATE>* because *to drown* is a hyponym of *to die*. We could also take into account more complex relations to find sentences like *The death of Frederick I Barbarossa on <DATE> led to ...* by recognizing the connection between *to die* and *death*. Harabagiu et al. proposed and evaluated a semantic matching solution [24] based on the WordNet, while Moldovan and Novischi proposed an algorithm for computing the strength of a relation between different concepts in [33]. An interesting solution is presented in [22], where a whole sentence is transformed into a semantic network, called *MultiNet*, and then matched to a question. Besides the WordNet matching, other important semantic connections of the question and the sentence are outlined in [34], such as similar noun phrases and co-occurring named entities.

Probabilistic matching is based on gathering a set of *features*, characterizing a similarity between the question and a candidate sentence, but also including other evidence, such as presence of named entities or information from WordNet. To assess importance of particular features and to select the sentence which is most likely to contain the answer, tools from the statistics are used, such as the logistic regression [35] or the entropy [36].

The last approach (conversion into logic forms) is less popular, but also has its advantages. The source text is converted into a formal language (say, a set of predicates) by a specialised parser, and knowledge obtained in that way may be used to find an answer in a process similar to theorem proving. Of course, once we have valid knowledge represented as a set of predicates, it is guaranteed that the answers will be valid, too. The only problem is that converting an ambiguous, imprecise, contextual and implicit natural language statements into strict logics poses a great challenge (if is possible at all). Apart from the already mentioned solution by Katz [13], an interesting system has been presented in [37]. To cope with those unpleasant features of a natural language, they propose a set of sources of world knowledge (common sense), such as lexical chains between concepts in WordNet, mentioned previously.

3.3.5 Answer extraction

If it is enough to give a sentence as an answer, the result of the selection outlined above (i.e. the best sentence) is presented to the user. However, if we want to show only an entity demanded, as proposed in section 2.3, then an appropriate extraction is necessary. Usually it is easy to select words corresponding to an entity of an expected type (e.g. a date). If there are more than one, we need to know from the previous stage, which of the entities has been matched as the answer. Providing an answer in a way proposed in section 2.3 p. 4 would involve the Natural Language Generation (NLG) capabilities.

3.3.6 Other modules

Apart from the stages described above, a question answering system may include a number of additional tools, improving its performance. Usually they are employed in the sentence selection stage.

- **WordNet** lexical database, used in a variety of roles, described above.
- **Named Entity Recognition (NER)** solutions are used to scan the text efficiently and find fragments which correspond to named entities[8]. They are useful in QA to reduce a number of sentences taken into account by processing only those containing named entities of type compatible with the question target and to extract the desired entity from the sentence.
- **Anaphora resolution** - anaphora, as described previously, is a word (usually a pronoun) substituting an expression which has already been mentioned. The process of resolution, i.e. assigning a set of possible targets to each anaphoric expression is very helpful in question answering. For example, let us consider answering the question *When did Albert Einstein publish special relativity theory?* The Wikipedia article contains the following:

  *On 30 April **1905**, Einstein completed his thesis, with Alfred Kleiner, Professor of Experimental Physics, serving as pro-forma advisor. Einstein was awarded a PhD by the University of Zurich. His dissertation was entitled "A New Determination of Molecular Dimensions". **That same year**, which has been called Einstein's annus mirabilis (miracle year), he published four groundbreaking papers, on the (...), special relativity, and (...), which were to bring him to the notice of the academic world.*[9]

  As we can see, the question target (year 1905) is three sentences away from the mention of the special relativity publication. The only link between them is the anaphoric expression *That same year*. Moreover, it does not refer to the whole named entity *30 April 1905*, but only to the year, which has to be taken into account during resolution (special relativity was published in 1905, but not on April 30). An example of successful incorporation of an anaphora resolution module into a QA system is presented in [5].

4 Prospects for Polish

One of the goals of this paper is to outline the prospects for building a Polish question answering system. Unfortunately, all of the solutions described above were prepared for English or German, which differ substantially from Polish in their structure. The purpose of this section is to analyse which of the elements of the question answering environment are directly applicable to Polish, which require some modifications, and which are hardly usable.

[8] The meaning of a named entity term is somewhat vague. Usually it is supposed to include entities referred by proper names (persons, organisations, countries, cities, mountains, lakes, etc.), titles of works (books, pieces of music, paintings, journals etc.), temporal (time, date, year, century) and numerical expressions (count, quantity, percentage, money).

[9] Extracted from the article in Wikipedia: `http://en.wikipedia.org/wiki/Albert_Einstein`.

4.1 NLP of Slavonic languages

Polish is a Slavonic language and shares most of the features common for the group. The most important ones from the point of view of NLP are: the rich nominal inflection, the free word order and the complicated inflection of proper names (both Polish and foreign). Herein, they are briefly explained only; a detailed review may be found in [38].

The nominal inflection makes text processing substantially harder, as every noun or adjective may take a number of forms[10] depending on its role in a sentence. On the other hand, it allows words to appear in different order without a complete change of meaning, as information about their roles in the sentence (e.g. distinction between a subject and an object of a verb) is preserved in the nominal inflection. For example, let us consider the following sentence: *John ate a fish*. Its obvious translation to Polish would be: *Jan zjadł rybę*. However, we could also say *Rybę zjadł Jan* with the same meaning but a slightly different focus. The remaining options (*Zjadł rybę Jan*, *Zjadł Jan rybę*, *Jan rybę zjadł*, *Rybę Jan zjadł*) would seem awkward, but remain perfectly understandable. This is not the case in English: *A fish ate John* has a totally different meaning. Of course, for longer sentences only some of the orderings preserve its comprehensibility; that is why this phenomenon is called the **relatively** free word order.

The nominal inflection affects also proper names, altering their appearance. An inflected form depend on numerous factors, such as a source language, a name pronunciation and a language tradition. The nominal inflection of foreign proper names confuse even native speakers of Polish; it is a cause of very common errors. Polish proper nouns inflect differently than their common equivalents too, for details see [38]. That is why the NER task is much harder in the Slavonic languages than in English.

4.2 Existing solutions for Polish

Only a few attempts to build a QA system have been made for Polish. First one, from 1985, was a Polish interface [39] to the ORBIS database, containing information about the solar system planets encoded as PROLOG rules. A POLINT system, based on it, is still being developed in the Adam Mickiewicz University in Poznań [40]. In 2002 another system capable of handling questions in Polish operating in a restricted domain (business information) was presented by Duclaye et al. [41]. Some elements (a reasoning module responsible for spatial relations) of a Polish QA system under development (*Hipisek.pl*) have been published by Walas [42].

A solution closest to what is analysed in this paper is that proposed by Piechociński and Mykowiecka [43]. It answers questions in Polish (translated from the TREC QA task) by scanning the content of the Polish Wikipedia. Unfortunately, it relies heavily on its structure, so couldn't operate on an arbitrary text corpus.

[10] In Polish there exist 7 cases, but in each declension some of them correspond to identical forms.

To sum up, several QA systems for Polish exist, but their usage is limited either by a reliance on a specific knowledge base [43] or by a closed domain (the remaining ones). None of them would be able to participate in a TREC competition in Polish, i.e. answer natural language questions using a given corpus of texts.

4.3 Language-dependence of elements

To discuss the possibility of building a Polish QA system, solving TREC-like tasks, the elements of the common architecture outlined in section 3.3 are analysed here to show their dependence on a processed language.

4.3.1 Text base processing

The main objective of this stage is to analyse words in order to replace them by a stem or a base form, common for all inflected forms of a particular word. In English it is done by stemming, i.e. transforming each segment according to a set of predefined rules. Unfortunately, because of the rich nominal inflection, the mutability of Slavonic words is much higher. Although stemmers for Polish exist as well [44], their performance is not very good. It is more recommendable to use a full morphological analyser, which processes a text slower, but guarantees to find every possible base form for each recognized segment.

4.3.2 Index

Index could work equally well with English and the Slavonic languages, if implemented properly. For example, the *Lucene* works perfectly with Polish corpora.

4.3.3 Question processing

The question processing module clearly needs to be prepared for a particular language, as question syntaxes may differ a lot between languages.

4.3.4 Sentence selection

Sentence selection method deserves special attention. Some of the approaches may depend on a processed language, but not necessarily all of them.

Let us consider sentence selection by pattern matching. It may be successfully applied to some languages, but Slavonic unfortunately will lead to much lower accuracy. That is because of the relatively free word order, which makes a pattern match only a most common ordering and ignore remaining ones. For example, let us try to answer the question *Who ate a fish?*. It would be converted to the pattern {<PERSON> <eat>[11] <fish>} and result in a match after finding the sentence *John ate a fish*. For Polish, the same question *Kto zjadł rybę?* correspond to the pattern {<PERSON> <jeść> <ryba>}, which matches the sentence *Jan zjadł rybę*, but fails in case of *Rybę zjadł Jan*, which carries the same meaning. This feature of the Slavonic languages also makes conversion into logic forms harder.

[11] By <eat> I mean any word which becomes identical to a base form *eat* after stemming, i.e. *eat*, *ate*, *eaten*, etc.

4.3.5 Answer extraction

Answer extraction stage depends heavily on a sentence structure, which is language dependent. This is another element, which becomes harder to implement for languages with the free word order.

4.3.6 Other modules

- **WordNet** contains relations about words in a particular language. Some of them correspond to relations between real-world entities (*a horse* is a hyponym of *an animal* in most languages), but some do not.
- **Named Entity Recognition (NER)** is an example of a task which becomes substantially more complex when implemented to the Slavonic languages. A core part of most such solutions is a *gazetter* i.e. a large database of known proper names. For example, we could most likely find there the name *Nelson*. If it appears in a text, NER module spots it easily. Unfortunately, this is not the case in Polish because of the rich nominal inflection - this very same name could appear in 5 different singular forms (*Nelson*, *Nelsona*, *Nelsonowi*, *Nelsonem*, and *Nelsonie*) and 5 plural (e.g. while talking about Nelson family). Complicated and often incorrect inflection of foreign proper names also needs to be taken into account.
- **Anaphora resolution** clearly depends on a processed language.

4.4 Linguistic tools

In the previous section we've mentioned a lot of linguistic tools which are relevant when building a QA system, but could not be used in different languages. Let us check their availability for Polish:

- **stemming** - As stated previously, several stemmers for Polish, discussed in [44], exist.
- **morphological analysis** - There are several tools of that type, *Morfeusz SGJP* [45], *Morfologik* [46], *PoliMorf* [46], to name only a few.
- **tagging** - Several taggers, capable of selecting the most probable interpretation of these provided by morphological analysers, exist, such as *PANTERA* [47] (using transformation-based learning) or *TaKIPI* [48] (using decision trees),
- **shallow parsing** - *Spejd* [49] allows to obtain a shallow structure of a sentence.
- **deep parsing** - *Świgra* [50] creates deep parsing trees in the DCG formalism.
- **named entity recognition** - Two tools could be employed for this task: *NERF* [51] and *Liner2* [52].
- **WordNet** - Widely available are: *plWordNet* [53], containing over 110000 synsets and *PolNet* [54] with approximately 11700 synsets.
- **anaphora resolution** - Polish anaphora resolution module (*Project CORE*) is under development [55].

As visible above, there are plenty of linguistic tools available for Polish, making a good starting point for developing a Polish question answering system.

5 Conclusion

In this paper, the possibility of building a Polish question answering system was discussed. First, a general explanation of a QA problem and possible improvements of the user experience by implementing natural language communication modules were shown to justify an effort of tackling the task. To focus on the most important properties of the QA problem, its definition, also usable for preparing an evaluation, was specified. Numerous solutions for English were presented to show their common architecture and distinctive features. Presented description of properties of the Slavonic languages and an analysis of language dependence of the elements of the scheme substantiate the claim that we need to apply a different approach to several issues, when Polish is concerned. Its feasibility is further supported by an enumeration of existing implementations of the relevant linguistic tools.

I demonstrated that an open-domain text-based question answering system for Polish is not only desirable but also achievable given the current level of development of Polish NLP. This study is a first step of building such a system.

Acknowledgements

Critical reading of the manuscript by Agnieszka Mykowiecka is gratefully acknowledged. Study was supported by research fellowship within "Information technologies: research and their interdisciplinary applications" agreement number POKL.04.01.01-00-051/10-00.

References

[1] Yandex.ru: Report on Yandex.ru queries. Technical report (2010)

[2] Hook, D.: Study of Search Engine Transaction Logs Shows Little Change in How Users use Search Engines. Evidence Based Library and Information Practice **1**(3) (2006) 88–94

[3] Andrenucci, A., Sneiders, E.: Automated Question Answering: Review of the Main Approaches. In: Third International Conference on Information Technology and Applications (ICITA'05). (2005) 514–519

[4] Shapiro, S.C.: Encyclopedia of Artificial Intelligence. John Wiley & Sons, Inc., New York (1992)

[5] Ferrández, A., Vicedo, J.L.: Coreference In Q&A. In Strzalkowski, T., Harabagiu, S.M., eds.: Advances in Open Domain Question Answering (Text, Speech and Language Technology). Springer Netherlands (2007) 71–96

[6] Ferrucci, D.A., Brown, E., Chu-carroll, J., Fan, J., Gondek, D., Kalyanpur, A.A., Lally, A., Murdock, J.W., Nyberg, E., Prager, J., Schlaefer, N., Welty, C.: Building Watson: An Overview of the DeepQA Project. AI Magazine **31**(3) (2010) 59–79

[7] Dang, H.T., Kelly, D., Lin, J.: Overview of the TREC 2007 Question Answering track. In: Proceedings of The Sixteenth Text REtrieval Conference, TREC 2007. (2007)

[8] Dang, H.T.: Overview of the TAC 2008 Opinion Question Answering and Summarization Tasks. In: Proceedings of Text Analysis Conference (TAC 2009). (2008)

[9] Peñas, A., Hovy, E.: QA4MRE: Question Answering for Machine Reading Evaluation at CLEF 2012 (2012)

[10] Sasaki, Y., Lin, C.j., Chen, K.h., Chen, H.h.: Overview of the NTCIR-6 Cross-Lingual Question Answering (CLQA) Task. In: Proceedings of NTCIR-6 Workshop Meeting. (2007)

[11] Ferrucci, D.A., Nyberg, E., Allan, J., Barker, K., Brown, E., Chu-Carroll, J., Ciccolo, A., Duboue, P., Fan, J., Gondek, D., Hovy, E., Katz, B., Lally, A., Mccord, M., Morarescu, P., Murdock, B., Porter, B., Prager, J., Heights, Y.: Towards the Open Advancement of Question Answering Systems. Technical report, IBM (2009)

[12] Simmons, R.F.: Natural Language Question-Answering Systems: 1969. Communications of the ACM **13**(1) (1970) 15–30

[13] Katz, B.: Annotating the World Wide Web Using Natural Language. In: Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet RIAO 97. (1997)

[14] Voorhees, E.M.: The TREC-8 Question Answering Track Report. In: Proceedings of The Eight Text REtrieval Conference (TREC 2000). Volume 7., National Institute of Standards and Technology (1999)

[15] Voorhees, E.M.: Overview of the TREC-9 Question Answering Track. In: Proceedings of The Ninth Text REtrieval Conference (TREC 2000). (2000)

[16] Voorhees, E.M.: Overview of the TREC 2001 Question Answering Track. In: Proceedings of the Tenth Text Retrieval Conference (TREC 2001). (2001)

[17] Voorhees, E.M.: Overview of the TREC 2002 Question Answering Track. In: Proceedings of The Eleventh Text REtrieval Conference (TREC 2002). (2002)

[18] Voorhees, E.M.: Overview of the TREC 2003 Question Answering Track. In: Proceedings of the Twelfth Text REtrieval Conference (TREC 2003). Volume 142. (2003)

[19] Voorhees, E.M.: Overview of the TREC 2004 Question Answering Track. In: Proceedings of The Thirteenth Text REtrieval Conference (TREC 2004). (2004)

[20] Voorhees, E.M., Dang, H.T.: Overview of the TREC 2005 Question Answering Track. In: Proceedings of The Fourteenth Text REtrieval Conference (TREC 2005). (2005)

[21] Dang, H.T., Lin, J., Kelly, D.: Overview of the TREC 2006 Question Answering Track. In: Proceedings of The Fifteenth Text REtrieval Conference (TREC 2006). Volume 2004. (2006)

[22] Hartrumpf, S.: Question Answering using Sentence Parsing and Semantic Network Matching. In: Proceedings of the 5th conference on Cross-Language Evaluation Forum: Multilingual Information Access for Text, Speech and Images CLEF'04ultilingual Information Access for Text, Speech and Images CLEF'04. (2004) 512–521

[23] Hovy, E., Hermjakob, U., Lin, C.Y.: The Use of External Knowledge in Factoid QA. In: Proceedings of the Tenth Text Retrieval Conference (TREC 2001). (2001)

[24] Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Gîrju, R., Rus, V., Morarescu, P.: The role of lexico-semantic feedback in open-domain textual question-answering. In: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics - ACL '01. (July 2001) 282–289

[25] Hovy, E., Gerber, L., Hermjakob, U., Lin, C.Y., Ravichandran, D.: Toward Semantics-Based Answer Pinpointing. In: Proceedings of the first international conference on Human Language Technology research (HLT). (2001)

[26] Fellbaum, C.: WordNet: An Electronic Lexical Database. The MIT Press (1998)

[27] Ahn, D., Jijkoun, V., Mishne, G., Müller, K., De Rijke, M., Schlobach, S.: Using Wikipedia at the TREC QA Track. In Voorhees, E.M., Buckland, L.P., eds.: Proceedings of The Thirteenth Text REtrieval Conference (TREC 2004). (2004)

[28] Li, X., Roth, D.: Learning Question Classifiers. In: Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002). Volume 1 of COLING '02. (2002)

[29] Buchholz, S., Daelemans, W.: SHAPAQA: Shallow Parsing for Question Answering on the World Wide Web. In: Proceedings of Euroconference on Recent Advances in Natural Language Processing RANLP2001. (2001)

[30] Ravichandran, D., Hovy, E.: Learning surface text patterns for a Question Answering system. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02. (2002) 41–47

[31] Duclaye, F., Yvon, F., Collin, O.: Learning paraphrases to improve a question-answering system. In: Proceedings of the 10th Conference of EACL Workshop Natural Language Processing for Question-Answering. (2003)

[32] Miliaraki, S., Androutsopoulos, I.: Learning to identify single-snippet answers to definition questions. In: Proceedings of the 20th international conference on Computational Linguistics - COLING '04. (August 2004)

[33] Moldovan, D., Novischi, A.: Lexical chains for question answering. In: Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002). (2002)

[34] Boni, M.D., Manandhar, S.: The Use of Sentence Similarity as a Semantic Relevance Metric for Question Answering. In: Proceedings of the AAAI Symposium on New Directions in Question Answering. (2003)

[35] Ko, J., Si, L., Nyberg, E.: A Probabilistic Framework for Answer Selection in Question Answering. In: Proceedings of the 2007 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics HLT-NAACL 2007, Association for Computational Linguistics (2007) 524–531

[36] Ittycheriah, A.: A statistical approach for open domain question answering. In Strzalkowski, T., Harabagiu, S.M., eds.: Advances in Open Domain Question Answering (Text, Speech and Language Technology). Volume 32 of Text, Speech and Language Technology. Springer Netherlands (2007) 35–69

[37] Moldovan, D., Paşca, M., Surdeanu, M.: Some Advanced Features of LCC's PowerAnswer. In Strzalkowski, T., Harabagiu, S.M., eds.: Advances in Open Domain Question Answering (Text, Speech and Language Technology). Volume 32. Springer Netherlands (2006) 3 – 34

[38] Przepiórkowski, A.: Slavonic information extraction and partial parsing. In: Proceedings of the Workshop on Balto-Slavonic Natural Language Processing Information Extraction and Enabling Technologies - ACL '07. (2007)

[39] Vetulani, Z.: PROLOG Implementation of an Access in Polish to a Data Base. In: Studia z automatyki, XII. (1988) 5–23

[40] Vetulani, Z.: Question answering system for Polish (POLINT) and its language resources. In: Proceedings of the Question Answering - Strategy and Resources Workshop, Las Palmas de Grand Ganaria, 28th May, Paris, ELRA (2002) 51–55

[41] Duclaye, F., Sitko, J., Filoche, P., Collin, O.: A Polish Question-Answering System for Business Information. In: BIS 2002, 5th International Conference on Business Information Systems, Poznań, Poland, 24-25 April 2002. (2002) 209–212

[42] Walas, M.: How to answer yes/no spatial questions using qualitative reasoning? In Gelbukh, A., ed.: 13th International Conference on Computational Linguistics and Intelligent Text Processing. (2012) 330–341

[43] Piechociński, D., Mykowiecka, A.: Question answering in Polish using shallow parsing. In Garabík, R., ed.: Computer Treatment of Slavic and East European Languages: Proceedings of the Third International Seminar, Bratislava, Slovakia, VEDA: Vydava- tel'stvo Slovenskej akadéme vied (2005) 167–173

[44] Weiss, D.: A Survey of Freely Available Polish Stemmers and Evaluation of Their Applicability in Information Retrieval. In: 2nd Language and Technology Conference, Poznań, Poland. (2005) 216–221

[45] Woliński, M.: Morfeusz — a Practical Tool for the Morphological Analysis of Polish. In Kłopotek, M., Wierzchoń, S., Trojanowski, K., eds.: Intelligent Information Processing and Web Mining. (2006) 511–520

[46] Woliński, M., Miłkowski, M., Ogrodniczuk, M., Przepiórkowski, A., Szałkiewicz, L.: PoliMorf: a (not so) new open morphological dictionary for Polish. In: Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012. (2012) 860–864

[47] Acedański, S.: A morphosyntactic Brill Tagger for inflectional languages. In: Proceedings of the 7th international conference on Advances in Natural Language Processing (IceTAL'10). (August 2010) 3–14

[48] Piasecki, M.: Polish Tagger TaKIPI: Rule Based Construction and Optimisation. Task Quarterly **11**(1-2) (2007) 151–167

[49] Przepiórkowski, A.: Powierzchniowe przetwarzanie języka polskiego. Akademicka Oficyna Wydawnicza EXIT, Warszawa (2008)

[50] Woliński, M.: An efficient implementation of a large grammar of Polish. Archives of Control Sciences **15** (2005) 251–258

[51] Savary, A., Waszczuk, J.: Narzędzia do anotacji jednostek nazewniczych. In: Narodowy Korpus Języka Polskiego [Eng.: National Corpus of Polish]. (2012) 225–252

[52] Marcińczuk, M., Janicki, M.: Optimizing CRF-based Model for Proper Name Recognition in Polish Texts. In: CICLing 2012, Part I. (2012) 258–269

[53] Maziarz, M., Piasecki, M., Szpakowicz, S.: Approaching plWordNet 2.0. In: Proceedings of the 6th Global Wordnet Conference. (2012)

[54] Vetulani, Z., Kubis, M., Obrębski, T.: PolNet – Polish WordNet: Data and Tools. In: Proceedings of the seventh International conference on Language Resources and Evaluation (LREC 2010). (2010)

[55] Kopeć, M., Ogrodniczuk, M.: Creating a Coreference Resolution System for Polish. In: The eighth international conference on Language Resources and Evaluation (LREC). (2012)