SQAD: Simple Question Answering Database

Marek Medveď and Aleš Horák

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xmedved1,hales}@fi.muni.cz

Abstract. In this paper, we present a new free resource for comparable Czech question answering evaluation. The Simple Question Answering Database, SQAD, contains 3301 questions and answers extracted and processed from the Czech Wikipedia. The SQAD database was prepared with the aim of a precision evaluation of automatic question answering systems. Such resource was currently not available for the Czech language. We describe the process of SQAD creation, processing of the texts by automatic tokenization (Unitok) and morphological disambiguation (Desamb) and successive semi-automatic cleaning and post-processing.
We also show the results of a first version of Czech question answering system named SBQA (syntax-based question answering).

Keywords: question answering, Simple Question Answering Database, SQAD, syntax-based question answering, SBQA

1 Introduction

The question answering (QA) field has a significant potential nowadays. Systems that are capable of answering any possible question can be widely used by general public to answer questions about the weather, public transport etc. In specialized applications, question answering can be used in medicine to speed up the process of fitting the patient symptoms to all possible illnesses.

Nowadays, several perspective question answering systems appeared, such as the IBM's Watson [1,2] that had a big success in popular television competitive show Jeopardy![1]. Many QA systems are orientated on a specific domain and are always able to answer only simple questions [3,4,5,6].

Question answering systems employ tools that process the input question than go through a knowledge base and provide a reasonable answer to the question. The presented SQAD database will help to measure and improve accuracy of QA tools as it offers all relevant processing parts, i.e. the source text, the question and the expected answer.

[1] Jeopardy! is an American television show that features a quiz competition in which contestants are presented with general knowledge clues in the form of answers, and must phrase their responses in question form.

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 121–128, 2014.

Original text:
Létající jaguár je novela spisovatele Josefa Formánka z roku 2004.
[Létající jaguár is a novel of writer Josef Formánek form the 2004.]
Question:
Kdo je autorem novely Létající jaguár?
[Who is the author of the novel of Flying jaguar?]
Answer:
Josef Formánek
URL:
`http://cs.wikipedia.org/wiki/L%C3%A9taj%C3%ADc%C3%AD_jagu%C3%A1r`
Author:
chalupnikova

Fig. 1: Example of a SQAD record

2 The SQAD Database

The Simple Question Answering Database, SQAD, was created in a computational linguistic course by students. Their task was to choose sections from Czech Wikipedia articles as a source for different questions and the respective answers. SQAD contains 3301 records with the following data fields (see Figure 1 for an example):

- the original sentence(s) from Wikipedia
- a question that is directly answered in the text
- the expected answer to the question as it appears in the original text
- the URL of the Wikipedia web page from which the original text was extracted
- name of the author of this SQAD record

In the first phase, the SQAD database consisted of plain texts. To support the comparison and development of question answering systems including SBQA [7] that is presented further in the text, SQAD was supplemented with automatic morphological annotations. The texts were processed with two tools: Unitok [12] for text tokenization and Desamb [8] morphological tagger, which provides unambiguous morphological annotation of tokenized texts (see Figure 2). Both tools are automatic systems and their accuracy is not 100% thus they occasionally make mistakes. To obtain high-quality data, the tagged texts were checked and corrected by semi-automatic and manual adjustments that are described in the following sections.

3 Adjustments of the SQAD Database

In this section, we describe amendments to the tokenization and tagging as obtained by Unitok and Desamb processing. Some of the modifications were

```
<s>
Kdo       kdo     k3yRnSc1
je        být     k5eAaImIp3nS
autorem   autor   k1gMnSc7
novely    novela  k1gFnSc2
Létající  létající k2eAgMnSc1d1
jaguár    jaguár  k1gMnSc1
<g/>
?         ?       kIx.
</s>
```

Fig. 2: Morphological annotation (consists of form, lemma and tag) of a sentence "*Kdo je autorem novely Létající jaguár? (Who is the author of the novel of Flying jaguar?)*"

a)
```
<s>
1
200
300
</s>
```
$\longrightarrow$ b)
```
<s>
1 200 300
</s>
```

Fig. 3: `Unitok`: a) wrong and b) adjusted tokenization of number "*1 200 300*".

systematic and proceeded in bulk over all the records, some of them were part of manual checking.

3.1 Tokenization Adjustments

During the manual checking of `SQAD` records, we found that large numbers have wrong tokenization if they contained whitespace as a thousands separator (see Figure 3). Therefore we have adapted the matching patterns that `Unitok` uses for processing the text to fix this setup.

3.2 Out-of-Vocabulary Words

The system `Desamb` is used for morphological tags disambiguation according to the word context. In some cases, the context is too narrow or there is no context at all so `Desamb` cannot determine the correct tag. In this case, `Desamb` returns "*k?*" (unknown kind) as the resulting tag. In `SQAD`, this is specifically true for the cases, where the answer in the `SQAD` record contains only a number. Then the `Desamb` resulting tag is "*k?*". Therefore we have systematically changed such tags to the "*k4*" tag for *numerals* (see Figure 4 for an example).

The `Desamb` tool is working over the attributive Czech tagset of the Majka system [9,10]. `Majka` is based on a deterministic acyclic finite state automaton

```
<s>                          <s>
120   120 k?       ⟶        120   120 k4
</s>                         </s>
```

Fig. 4: `Desamb`: unrecognized number

```
<s>                                  <s>
Los      Los      k?                 Los      Los      k1
Angeles  Angeles  k?                 Angeles  Angeles  k1
</s>                        ⟶        </s>
<s>                                  <s>
LA       LA       k?                 LA       LA       kA
</s>                                 </s>
```

Fig. 5: `Desamb`: unrecognized proper names and abbreviations

(DAFSA) that is created from large morphological database. Despite the coverage of more than 30 million Czech verb forms, `Majka` does not recognize all the existing words, especially proper names and abbreviations. Thus the resulting tag on such words is usually "*k?*", which means that the system does not contain this word in the database and the context is not long enough to guess the tag. For such words, we have systematically changed the *unknown* tag to:

a) "*k1*" (nouns) for all words that start with an upper case letter, and
b) "*kA*" (abbreviations) for words that contains only upper case letters, words ending with dot or words containing dots between upper case letters.

See Figure 5 for the resulting annotation of unknown proper names and abbreviations.

As we have mentioned above, the `SQAD` database is extracted from Wikipedia texts, which are multilingual and the texts often contain original forms of proper names. For example original writing of the name "*Tokio*" is "東京". These foreign language words are not included in `Majka` database thus `Desamb` always assigns them the tag for unknown words ("*k?*").

To fix the remaining unknown word tags in `SQAD`, we have extracted all unknown words into one file keeping the original file name, word position and unknown word with its lemma and tag from `Desamb`. This file was than manually annotated and programmatically applied back to the original annotated file (see Figure 6).

3.3 Mistakes in Morphological Analysis

In case of unknown words, the `Desamb` tool can not only leave the word tag undecided, but if the unknown word has a suitable context, `Desamb` can "guess" its lemma and tag even if the word is not present in the `Majka` database. This

Original `Desamb` output stored in file `03text.txt`:

```
Tokio Tokio k1gInSc1
(     (     kIx(
jap.  jap.  kA
東京  東京  k?
```

Record of unknown word extracted from `03text.txt` file:

```
./0000/03tex.txt|3|東京  東京  k?
```

Record from `03text.txt` with manual changes:

```
./0000/03tex.txt|3|東京  東京  k1
```

File `03text.txt` with changes:

```
Tokio Tokio k1gInSc1
(     (     kIx(
jap.  jap.  kA
東京  東京  k1
```

Fig. 6: `Desamb`: unrecognized foreign words

works very well when `Desamb` processes Czech words, however it may cause mistakes for foreign words. In this case not only the tag is wrong but also the lemma. For example if we have word *"Las"* (from proper name *"Las Vegas"*) the output of `Desamb` is "Las laso k1gInSc1" (where word *laso* means *lasso*).

To repair all these mistakes, we checked all the `SQAD` database records and extracted a similar file as in Section 3.2 and corrected the tags in the original `SQAD` files.

4 Evaluation

We have used the `SQAD` database to evaluate the accuracy of a first version of `SBQA`, syntax-based question answering system [7]. `SBQA` was developed during a master thesis by Matej Pavla at Faculty of Informatics, Masaryk University. The input of `SBQA` system is a plain text question which is then preprocessed by `Unitok` and `Desamb` system and passed to `SET` [11] parser to identify dependencies and phrase relations within the question. `SBQA` then decides the rules for finding the answer in its knowledge base based on a match on corresponding syntactic structures (hence the "syntax-base" in its title).

The `SBQA` knowledge base is made also from plain text documents, which are automatically processed with the same tool chain as the questions (`Unitok`, `Desamb` and `SET`). As we presented in this paper, the automatically preprocessed texts contain mistakes. For the purpose of evaluation of the QA part of `SBQA`, we have modified its workflow to build the knowledge base from the annotated `SQAD` database and not to use the automatic processors.

Table 1: Evaluation of SBQA system

total questions	correct	partially correct	incorrect	not found
3,301	758	60	2,003	480
100%	23%	1%	61%	15%

Table 2: Classification of SBQA errors (on 200 examples)

total questions	error in SBQA system	error in tokenization or syntax analysis	uncovered phenomena
200	119	43	38
100%	59.5%	24.5%	19%

For the original results see [7]. The current results of SBQA as measured on the SQAD database are presented in Table 1.

After the evaluation, we have taken 200 questions that are incorrectly answered by SBQA and manually checked the reason of the mistake. We have obtained three categories of mistakes that are caused by:

1. errors in implementation of SBQA system
2. errors in tokenization or syntactic analysis
3. phenomena not covered by the current implementation of SBQA system

An overview of the percentage of these error categories is presented in Table 2. A detailed description of particular error categories can be found in the following subsections.

4.1 Errors in Implementation of SBQA System

The SBQA system balances between answer correctness and the ability to find the answer. Because of that the SBQA implementation uses concept of choosing the answer candidates based on their probability values. On the other hand this implementation sometimes leads to a wrong evaluation that produces incorrect answer.

We have identified the following error types that are caused by SBQA implementation:

– answer in brackets: if the answer of the question is placed in brackets, the SBQA system nearly always provides incorrect answer
– part of speech requirement: even if the tokenization and syntax analysis is provided well, the SBQA system does not check the part of speech that is important for answering the question and the system answers incorrectly
– comparison of dates or numbers: in case of a question, whose answer needs to perform some date or numeric computations, the system does not provide the answer (only exact matches are found).

– wrong question type: as described in [7], SBQA determines for each question its question type. But sometimes the system provides an answer to the question with yes/no question type even if the original question is not a yes/no question.

4.2 Errors in Tokenization or Syntactic Analysis

The SBQA system answers incorrectly in case the tokenization or syntactic analysis contains mistakes.

There are three types of such errors that appear in the current SQAD database:

– Desamb incorrectly detects sentence boundaries and splits one sentence into two or more sentences
– Desamb incorrectly tagged a word thus the syntactic analysis is incorrect and SBQA system cannot derive the required answer
– SET incorrectly parses a sentence and creates an incorrect syntactic tree. This usually leads to incorrect answer.

4.3 Uncovered Phenomena

The SBQA system has not yet implemented advanced NLP techniques such as anaphora resolution. Which means that if the answer to the question is in the sentence that refers to previous text then the answer cannot be found.

5 Conclusions

In this paper we have presented new Czech question answering database called SQAD. Each SQAD record consists of an annotated question, the annotated answer, the annotated sentence containing the full answer, Wikipedia URL as a source of the statement and the author name of this question-answer pair. The morphological annotation was obtained automatically and manually corrected. The corrections make SQAD data more precise so they are more suitable for evaluation of Czech question answering systems.

The SQAD database in its current version is available for download at http://nlp.fi.muni.cz/projects/sqad.

Acknowledgements This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

References

1. Ferrucci, D.A.: IBM's Watson/DeepQA. SIGARCH Comput. Archit. News **39**(3) (2011) – http://doi.acm.org/10.1145/2024723.2019525.

2. Ferrucci, D.A.: IBM's Watson/DeepQA. In: Proceedings of the 38th Annual International Symposium on Computer Architecture. ISCA '11, New York, NY, USA, ACM (2011) – http://dl.acm.org/citation.cfm?id=2000064.2019525.
3. Krishnamurthi, I., Shanthi, P.: Ontology based question answering on closed domain for e-learning applications. Australian Journal of Basic and Applied Sciences **8**(13) (2014) 396–401
4. Chung, H., Song, Y.I., Han, K.S., Yoon, D.S., Lee, J.Y., Rim, H.C., Kim, S.H.: A practical QA system in restricted domains. In: Proceedings of the Workshop Question Answering in Restricted Domains, within ACL. (2004)
5. Vargas-Vera, M., Lytras, M.D.: Aqua: A closed-domain question answering system. Information Systems Management **27**(3) (2010) 217–225
6. Wang, D.: A domain-specific question answering system based on ontology and question templates. In: Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD), 2010 11th ACIS International Conference on, IEEE (2010) 151–156
7. Pavla, M.: Automatic question answering system. Master thesis, Masaryk university, Faculty of Informatics (2014) http://is.muni.cz/th/418138/fi_m/?lang=en.
8. Šmerk, P.: Unsupervised learning of rules for morphological disambiguation. In: Text, Speech and Dialogue, Springer Berlin Heidelberg (2004) http://dx.doi.org/10.1007/978-3-540-30120-2_27.
9. Šmerk, P.: Fast Morphological Analysis of Czech. Proceedings of Recent Advances in Slavonic Natural Language Processing (2009)
10. Jakubícek, M., Kovář, V., Šmerk, P.: Czech morphological tagset revisited. Proceedings of Recent Advances in Slavonic Natural Language Processing **20** (2011) 29–42
11. Kovář, V., Horák, A., Jakubíček, M.: Syntactic analysis using finite patterns: A new parsing system for czech. In: Human Language Technology. Challenges for Computer Science and Linguistics, Berlin/Heidelberg (2011) 161–171 http://dx.doi.org/10.1007/978-3-642-20095-3_15.
12. Michelfeit, J., Pomikálek, J., Suchomel, V.: Text Tokenisation Using unitok. In Horák, A., Rychlý, P., eds.: 8th Workshop on Recent Advances in Slavonic Natural Language Processing, Brno, Tribun EU (2014) 71–75