Grant Agreement Number: 257528

KHRESMOI

www.khresmoi.eu

Analyzed parallel corpus from the biomedical domain

Deliverable number	*D4.2*
Dissemination level	*Public*
Delivery date	*8 February 2012*
Status	*Final*
Author(s)	*Jakub Bystroň, Jan Hajič, Pavel Pecina*

This project is supported by the European Commission under the Information and Communication Technologies (ICT) Theme of the 7th Framework Programme for Research and Technological Development.

Executive Summary

This deliverable concludes the work on training a Czech/English Statistical Machine Translation system conducted at CUNI in Year 1 of Khresmoi in the multilingual workpackage (WP4). The main focus was put on acquisition and processing of in-domain training resources and optimization of the SMT engine for the biomedical domain. We did not succeed in acquiring an in-domain parallel corpus of a sufficient size because of very limited available resources (in fact no such resources were usable for Khresmoi) and used the CzEng corpus as a suitable alternative to train the translation models though it contains data from various different domains. However, for building language models, we used in-domain monolingual data automatically acquired from the web. We automatically crawled hundreds of thousands of web pages from web sites presumably containing Czech texts from the biomedical domain. Then, we extracted the textual content of the pages and identified relevant in-domain passages using a domain classifier specially trained for this purpose. As the result we obtained monolingual corpora from the domain of interest and used it to train domain specific language models for our SMT system. We also created an in-domain parallel corpus for parameter optimization and evaluation of our system (over one thousand of sentence pairs). All these resources were used to train our SMT system which was presented during the project review in Sierre. In this report we provide details of the corpus acquisition process, training the SMT system and its empirical evaluation. The systems are based on a phrase-based translation model trained with the open-source MT system Moses and the language model toolkit SRILM.

Table of Contents

List of Abbreviations

ANN:	Artificial Neural Networks
BLEU:	Bilingual Evaluation Understudy
BMČ:	Bibliographia Medica Čechoslovaca
CUDA:	Compute Unified Device Architecture
CUNI:	Charles University in Prague
GPU:	Graphics Processing Unit
IE:	Information Extraction
LM:	Language Model
MERT:	Minimum Error Rate Training
MeSH :	Medical Subject Headings
MT:	Machine Translation
SMT:	Statistical Machine Translation
SVM:	Support Vector Machines
ÚFAL:	Institute of Formal and Applied Linguistics, CUNI

1 Introduction

The most important contribution to the multilinguality work package in Khresmoi is the machine translation. Our first task was to collect data and prepare the pilot version of MT system with a focus on the biomedical domain of Khresmoi. After some research it turned out that there were no existing parallel in-domain biomedical Czech/English corpora at the time. Unfortunately, even web-sites focused on health with parallel language mutations were non-existent. We therefore decided to use our CUNI's parallel generic Czech/English corpus CzEng and to focus on obtaining the high-quality in-domain monolingual data.

We collected the required data, trained the Moses open-source statistical machine translation system and built a web demo which was presented at the project review in Sierre.

2 Statistical Machine Translation

2.1 Basic Concepts

Machine translation is complicated. Human languages are so complex and diverse that the task of translating a document from one language into another cannot be modeled by precise mathematical formulae perfectly.

We assume that sentences of a given document can be translated one by one. That is we do not use context information from the sentences translated before. This somewhat non-trivial restriction is nowadays totally common because context-aware SMT systems would need much more complex models which could be computationally infeasible on present hardware. It is also not clear how to exploit the context usefully. On the other hand, there are definitely blocks of sentences which are problematic to translate well without any broad context information. This fact can also differ between languages.

The cornerstone of Statistical Machine Translation can be seen in this equation:

$$\hat{e} = \arg\max_e P(e \mid f) = \arg\max_e P(e) \cdot P(f \mid e)$$

Simply said, the $P(f \mid e)$ on the right-hand side is a probability mass function of the translation model and the $P(e)$ relies to the language model. The translation model is "responsible" for the word relations between two languages whereas the language model gives us the likelihood that a given sentence is a proper well-formed sentence in the target language. We note that hats everywhere (as usual) denote estimates and the letters ***e*** and ***f*** denote sentences in the target and source language respectively (abbreviated English and French sentences for historical reasons). Direction is always from ***f*** to ***e***. We also note that the above equation follows from the simplest form of Bayes theorem.

In our case we use so called **log-linear system**. The equation above can be then restated as

$$\hat{e} = \arg\max_e \sum_{i=1}^{M} \lambda_i h_i(e, f),$$

where h_i-s are the so-called *feature functions* and lambdas are *the weights* of the log-linear model which have to be estimated.

Any number of feature functions can be of the form

$$h(e, f) = h(e).$$

These feature functions are then called **a language model.** In our work we use only n-gram language models described in the text below. Any detail on SMT can be found in [1].

3 Parallel Data

3.1 Czech/English Issues

As mentioned above there were no parallel biomedical corpora at the time. We also tried but did not find any reasonably big parallel web-sites which could have been crawled. So we decided to use the parallel Czech/English generic corpus created at CUNI and to focus on monolingual data. On the other hand, this parallel corpus is quite big and of a high quality so we probably can claim that this did not affected the results of the final SMT system very much but, to be rigorous, we cannot prove or disprove this claim.

3.2 CzEng Corpus

CUNI has created a large parallel Czech – English corpus CzEng of high quality (see [4]). Work on CzEng started back in 2005 and the current version CzEng 1.0 has the parameters depicted below. We would like to emphasize that a parallel and deeply annotated corpus of this size is quite unique for smaller languages like Czech.

Statistics of CzEng 1.0 Sources (Table 1)

CzEng 1.0 Sources	Nr. of sentences	Tokens
JRC-Acquis EU	3,992,551	26 %
Subtitles	3,076,887	20 %
Books (fiction…)	4,335,183	29 %
Technical documentation	1,613,297	12 %
Parallel websites	1,883,804	13 %
TOTAL	14,901,722	201,413,571

CzEng is built from a bigger corpus and consists only of aligned and filtered good quality sentence pairs.

4 Monolingual Data

SMT system to be trained needs both parallel and monolingual data. Monolingual corpora are used to compute a probabilistic *language model* which orders a SMT system to produce sentences which are likely in the target language. Thus, the importance of monolingual data is very high.

4.1 Data Acquisition

Much work has been done on acquiring high-quality monolingual data especially in Czech where no in-domain (biomedical) corpora existed before Khresmoi. We found out that BMČ (Bibliographia Medica Čechoslovaca [20]) was the only resource usable out of the box of medical articles in Czech suitable for Khresmoi purposes at the time. Unfortunately, BMČ (as its name suggests) does contain only bibliographical entries (617,155 entries) with MeSH annotations (nice, used for multilingual IE in Khresmoi by J. Dědek, [21]). No abstracts or even full articles are in BMČ but some entries (precisely 25,408) contain links pointing at various places on the internet. There is no central repository of those articles perhaps because of licensing issues. Using approximately 40% of the links we were able to download the linked papers. Remaining links pointed to servers which were offline or did not provide these papers any more. The articles being accepted for publishing in medical journals are of excellent quality and did not need much filtering apart from encoding unification and elimination of unprintable characters by few scripts.

Based on this finding we therefore decided to prepare a crawler and to automatically crawl the biggest Czech sites focused on health and medicine. We manually picked 112 high-ranked seeds (typically root URLs of that sites) from the catalogues of the biggest web-search portals in the Czech Republic (Seznam [22], Centrum [23]) and let our crawler download everything reachable. There were many standard crawling issues, few of them we mention in the implementation notes.

The Biggest Crawled Sites and Portals (in Czech):

- http://zdravi.doktorka.cz/
- http://www.magazinzdravi.cz/
- http://www.medicina.cz/
- http://www.prolekare.cz
- http://www.abecedazdravi.cz/
- http://www.helpnet.cz/

Using the crawler we obtained a big (22.76 GB) raw meta-corpus consisting of raw HTML pages from the biomedical sites.

4.2 Preprocessing HTML

It was obvious that automatically crawled web-pages despite their origin on manually picked biomedical sites contained not only the relevant biomedical texts. Among other things they contained advertisements, licenses, chats and so on which we definitely did not want to have in our monolingual corpus. So we needed to filter out the out-domain parts. Since our crawler downloaded raw HTML web pages we needed to parse the content of HTML and to extract only the plain-text and in-domain parts. Moreover, we also tried to retain the text locality in the web-page.

We needed to get rid of HTML tags and non-textual content and to partition the textual content of web-pages into the *blocks* which could have been filtered afterwards. Extraction of the textual content from HTML was a simpler task because, simply said, it suffices to have a reasonably robust parser which parses even the malformed HTML (rather norm than exception on web sites). We note that even a valid HTML needs not to be a XML (tree). We used a custom built parser by one of the authors. We omit the details of this parser in this report.

To sketch the idea of the parser let us mention that from the one particular HTML page we could, in general, obtain several so called *blocks*. A block is a text-only excerpt from the original HTML got from "one place" which is a merged text from big enough sub-tree of the parse tree, e.g. from one paragraph, few (even nested) divs and so on. We set the lower limit to 64 words. More precisely, robust HTML parser tries to reconstruct the DOM tree. HTML on the internet usually is not XML so there is not a unique tree representation. In that DOM tree we identify paragraphs and bigger divs and merge the text content in them omitting HTML tags. This procedure gives us those blocks. We are sure that this specific task is very interesting and should be investigated on its own but our quite simple procedure was giving us satisfactory good results on our raw data. Precision compared with human judging on the random sample of 72 web-pages was 82%.

4.3 Morphological Analysis

After parsing all the downloaded stuff we merged everything together and created a big monolingual but still raw meta-corpus consisting blocks. We wanted to retain the block information (IDs) because our biomedical classifier classifies not the sentences but whole blocks. We guessed that it was a reasonable criterion to accept or deny a whole block because block contains much more information which can be exploited than only a sentence. Because it is not clear what a "biomedical block" is, it is not definitely easier what a "biomedical sentence" is.

We then tagged our meta-corpus with the perceptron-based tagger Morce built at UFAL ([18]). Its accuracy on PDT 2.0 ([24]) corpus is 96.1%.

4.4 Classification of Biomedical Content

We experimented, independently, with several standard Machine Learning algorithms (SVM, ANN…) to find out the best framework for our purposes. The best results (by a human decision) were obtained using Support Vector Machines. We used LibSVM (Chang, Lin [14]) for that. Experiments using Artificial Neural Networks were done using a very nicely implemented C library FANN (Fast Artificial Neural Network Library, [15]) but the results of SVM were better.

4.4.1 General Notes on SVM Classifier

Support Vector Machines (SVM) are the well established machine learning (ML) method with excellent potential to solve classification or even regression problems of many kinds. The biggest advantage of SVM is its strong mathematical background and the fact that the theory existed before SVM were invented (compare with ANN – ANN were used successfully for years without satisfactory theory). In SVM we can apply many tricks (not only the kernel one) and methods from e.g. functional analysis or linear algebra as its special case. The main idea of SVM (in its simplest case) is to find a hyperplane in the high-dimensional Hilbert space (normed linear space with the norm induced from the inner product, moreover complete) which separates (with the least possible error) positive and negative examples and simultaneously maximizes the distance to the positive and negative examples (points in that space). This task is solved by standard algorithms from convex optimization.

One of the most beautiful things in machine learning is that with SVM we can even map examples (both positive and negative) to the infinity-dimensional Hilbert space, and yet the training and querying can be done in perfectly finite time. Mapping to higher dimensional spaces using so called kernel-trick is standard because one has typically much higher chance to find a good separating hyperplane in higher dimension.

Let us conclude this note on ML with the claim that SVMs are much more complicated if they are to be used in practice. From the computational point of view (and if we have corpora of size in tens of GBs it is a must) the biggest problem is just the big dimensionality and so it is desirable to find ways how to map the original space to some lower-dimensional space with "low distortion" (feature extraction, PCA, randomized mappings…).

4.4.2 SVM Classifier

The input of our classifier was the above mentioned meta-corpus consisting of blocks extracted from HTML pages. Each block was endowed with its unique ID and it was lemmatized and stop words (see the Appendix 10.2) were removed by the custom list-based hand-written stop word eliminator. This form of a block we can call *standard*. We used so called *Supervised Machine Learning* to train our SVM stack. We trained our SVM using in-domain blocks obtained from the in-domain monolingual data. Just normal size-split sufficed to obtain these blocks which we considered as positive examples. We mention that SVM, typically presented in its simplest form as a two-class classification algorithm, can be also trained with only positive data. See e.g. [25] for details.

Each block was represented as a feature vector. We repeat here that our blocks were tokenized, stop words were removed and each word was changed to its lemma. We then computed several unigram and n-gram features, e.g. TF, TF-IDF, Hadamard (vector, i-th entry is the product of the frequency of the i-th keyword and its frequency in the training set). We then used PCA to reduce the dimensionality of the training set. We tried polynomial, RBF (radial basis function) and sigmoid kernels and picked the RBF kernels. See [25], [26] for more details. There were two unknown parameters C (penalty parameter), T (RBF parameter) to be determined for the RBF kernel based SVM. They were estimated using the so-called *cross validation* as explained below.

4.5 Tuning and Evaluation

We optimized the two unknown parameters of the SVM classifier using so called *K-fold cross validation*. This technique is a classical statistical tool for predicting real world performance (precision) of many algorithms. We randomly partitioned the training set of biomedical blocks several times into $K = 5$ disjoint subsets. One subset were then picked to be the testing and it was cross-validated using SVM model trained on the others. This scheme is repeated with different partitionings in several rounds. Validation results are then averaged over all rounds.

The two unknown parameters were searched using a *grid-search* method in conjunction with the cross-validation. Grid-search method tries many pairs of unknown parameters from the matrix

{ 1/64, 1/32, 1/16, ..., 1, 2, 4, 8, ..., 32768 } x { 1/64, 1/32, 1/16, ..., 1, 2, 4, 8 }

of all possible pairs *(C, T)*. This method is straightforward but, more importantly, excellently parallelizable.

We tuned the classifier and obtained by methods mentioned above a classifier with precision 82% on our data. We must admit that this number can be hardly viewed as strictly rigorous because of the implicit uncertainty of the selection what are biomedical blocks and what are not. Even a human eye cannot always decide with certainty. But we hope that our classifier did its work properly.

4.6 Language model

Simply said, language model tells us how probable is a given sentence in the given (target) language, more precisely, it defines a probability distribution over sequences of words given data, $P(w_1,...,w_n \mid Data)$. We built several up to 5-gram English and Czech monolingual language models based on sentences from the fixed corpora acquired from various resources (automatically web-crawled data, data sets mentioned on the Khresmoi webpage).

We used a SRILM ([5]) tool with a Kneser-Ney smoothing for this. SRILM can be freely obtained on the internet and as a C application can be compiled with GNU C compiler gcc. In our case we used for training the language model a patched version 1.0.5 and tuned a little bit only a compiler arguments to get the fastest possible binaries (optimization levels, target architecture, SSE, inlining…).

This version at the time of writing this paper can be obtained here:

http://www.speech.sri.com/projects/srilm/

or with many Moses distributions.

Number of n-grams as produced by SRILM (Table 2)

n	Number of n-grams
3-gram	27,983,783
4-gram	280,240,743
5-gram	320,320,258

Let us note that the research area of language modeling is very active. Baseline models under the hood are typically the likelihood Markov models of word sequences probabilistically computed from n-gram counts from the language model corpus. Those estimates are then almost always smoothed, e.g. with Kneser-Ney smoothing. Precisely this setting was used to train our language models.

5 Training the SMT System

5.1 Collected Data (Input)

Let us now summarize the requirements on the data needed to train the pilot version of in-domain SMT system for Khresmoi using Moses. See [1] or [16] for details.

- **Parallel Data** – English and Czech, sentence aligned (n-th line of English version file corresponds to the n-th line in Czech version file).
    1. **Big Training Data (BTD) –** We used sentence aligned corpus (CzEng 1.0) with 14,901,722 sentences in Czech and English.
    2. **Held-out Data for Optimization –** Corpus of 1,212 manually translated sentences not contained in BTD. The last phase of the training process searches for the best values of several parameters that maximize some measure of the quality of translation on this corpus. We used a log-linear system with BLEU as the target measure and the parameters were tuned on this corpus.
    3. **Data for Testing –** Corpus of 956 manually translated sentences not contained in BTD.
- **Monolingual Data –** Target language data, 12,870,939 sentences, in-domain. The size of the English vocabulary (all word forms) was 108,072.

We thank the anonymous translator for translating all sentences of the items 2 and 3. The key point here is to emphasize that it is desirable to have both training and testing data from the same domain. Otherwise the quality of translation drops a little bit (see details in [17]) due to possible poor vocabulary coverage and different language specifics of diverse domains.

5.2 Moses

We used Moses ([11]) to train the phrase-based model using *train-model.perl* script (encapsulates GIZA++ [27] and *mkcls* calls) with several parameters specifying the parallel sentence-aligned corpus, target language model and the training options.

After that the parameters of log-linear models are determined using MERT with BLEU as the optimization measure (not metric). At this case we use a small in-domain excerpt of the parallel corpus on which we optimize the parameters (1,212 sentences in our case). These sentences as well as the testing parallel data (956 sentences) were manually translated at CUNI. The sentences were randomly picked from pre-chosen sentences of the length between 20 and 40 words from the monolingual English data. We hope these two corpora are of high-quality.

For each phrase pair in our case Moses computed five coefficients:

- Forward and backward phrase translation scores (based on normalized co-occurrence counting)
- Two lexical weights (language model scores)
- Constant penalty

We note that a log-linear model has the form

$$P(x) = \exp \sum_{i=1}^{n} \lambda_i h_i(x)$$

in the probability space of *(e, f, begin, end)* tuples where lambdas are weights and h_i-s are feature functions specified above. We chose the best lambdas, as mentioned above, using a so called minimum error rate training (MERT – Och, 2003) which is a type of coordinate descent method.

The whole training sequence can be summarized as follows:

1. **Tokenization** – Was done using Moses tokenizer scripts (default settings for Czech or English), on both parallel and monolingual data.
2. **Filtering** – Rule out parallel sentences shorter that 16 tokens or longer than 50.
3. **Lowercasing –** Machine translation systems are usually trained on lowercased data for space saving purposes. We use the Moses recaser script *train-recaser.perl* for training the recasing model. This script creates a simple translation model from the cased training corpus. All translations obtained from Moses are then recased using this model before presenting to the user.
4. **Building the language model** – Using SRILM (tunable, not straightforward) – we built several models (3, 4, 5-grams), finally only 5-grams were used, filtered sentences to the maximum length of 50 and the fertility ratio in [0.12, 8.0].
5. **Building the translation model, Recasing –** Moses scripts encapsulating GIZA++/mkcls calls
6. **Tune the weights in the log-linear system –** Using the small in-domain held-out parallel manually translated corpus (1,212 sentences)
7. **Evaluation –** Using the small in-domain held-out parallel manually translated corpus (956 sentences)

6 Evaluation and Conclusion

In Year 1 we have prepared the monolingual and the parallel multilingual corpora for Khresmoi (Deliverable 4.2, WP4). It turned out that collecting strictly in-domain parallel corpus was nearly impossible for a specific Czech/English language pair at the time. So we picked a general purpose CzEng corpus and focused our work on the obtaining high-quality in-domain monolingual data. We were successful in acquiring and filtering much monolingual data and prepared the in-domain monolingual corpus used then for the language model training.

The BLEU score we obtained with our SMT system was 0.28 on our testing data set. For details about the BLEU metric see [28]. Our next task will be to prepare SMT systems for other language pairs (EN/DE, EN/FR…). For these language pairs preparing the parallel and strictly in-domain corpora will be definitely a feasible task because much data already exists and much data has been already collected by CUNI. However let us underline that SMT systems for different language pairs albeit based on the same phrase-based model algorithms are totally different because of diversities in languages and available data. For example German compound words are one of the big challenges for phrase-aligning.

7 Notes on Implementation

7.1 Implementation, Hardware, Hacking

All the crawled data was acquired using our custom built high performance web-crawler written in the C (C99) language and compiled with GCC 4.6.1. This crawler is based upon the so called EEQ library of one of the authors. The main focus of this library is speed and low memory consumption and implements various things, e.g.:

- **Fast filtered and buffered I/O** – libc's stdio is quite old and rigid and doesn't provide many techniques which are crucial for high-performant systems such as online fast compression etc.
- **High speed compression** – because storage is nowadays rarely a bottleneck, in our code we use rather faster compression stacks than e.g. gzip or even bzip2. We have picked LZO [12] which is an implementation of Lempel-Ziv-Oberhumer algorithm with much faster decompression than any of the above mentioned algorithms (tools).
- **Data structures –** we have also reimplemented many libc routines (e.g. sorting) or basic data structures. This gave us nearly 30% speed advantage e.g. for hashing compared with JDK7 implementation.
- **SIMD Instructions –** because of the nature of many text algorithms we use in the crawler, classifier and so on one can harvest some additional speed by reimplementing various routines with vectorized instructions. Current CPUs have many instructions that operate with 16 B or even nowadays 32 B (bytes!) of data at a time. It is nothing new since MMX is an old and gray technology but it is still not used as it could be. Of course, C compilers (gcc, icc) with proper optimization flags are able to vectorize some code but not always. This gave us a speed-up like 1.5-2.5 for many routines.
- **Computation on GPU –** one area in which we experimented only a little bit is CUDA. Current state-of-the-art GPUs contain hundreds of simple cores and a super-fast in-place memory with approximate bandwidth around 130 GB/s or even more. Many textual algorithms are perfectly parallelizable and an ideal fit for computation on GPU. Reimplementation of those algorithms using the CUDA toolkit [13] is not straightforward and one must consider many factors like CPU-GPU memory transfers (the simplest) and so on.

  In this context we cannot resist the temptation to say that we were very disappointed by high-end NVIDIA's so called Tesla graphics units which are in the top product line intended for 24/7 server-side computation. We found that not only at CUNI these cards are very likely to

fail after few days of computation due to some unknown reasons. After a few days they very often suddenly started to give wrong results because of maybe some memory corruption. The only way to deal with this problem right now is to restart the hardware from time to time or to increase the voltage a little bit. Despite these problems we see in GPU computations (or in parallelization in general) a fantastic potential for the whole area because current state-of-the-art tools are not optimized well and researchers waste much time waiting for the results of their experiments.

- **Other stuff –** among many other things let us only mention a custom region-based memory allocator it gave us better results in algorithms doing many small allocations.

8 References

[1] Koehn, P.: Statistical Machine Translation. *Cambridge University Press (2010)*

[2] Och, F. J., Ney, H.: A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics, Volume 29 (2003)*

[3] Bojar, O., Prokopová, M.: Czech–English Word Alignment. *Technical Report UFAL MFF UK (2006)*

[4] Bojar, O., Žabokrtský, Z.: CzEng corpus. *UFAL MFF UK Technical Report (2010)*

[5] Stolcke, A.: SRILM - An Extensible Language Modeling Toolkit. I*n Proc. Intl. Conf. Spoken Language Processing, Denver, Colorado (2002)*

[6] Koehn, P.: Statistical Machine Translation, 1st ed. *Cambridge University Press (2010)*

[7] Goldwater, S. et al.: A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition 112 (2009)*

[8] Koehn, P.: Europarl: A multilingual corpus for evaluation of machine translation *(2002)*

[9] JSON: JavaScript Object Notation. *http://www.json.org/ (2012)*

[10] Lacoste, J. et al.: Word alignment via quadratic assignment. *In Proc. HLT-NAACL (2006)*

[11] Moses: Statistical Machine Translation System. *http://www.statmt.org/moses/*

[12] Oberhumer, M.: LZO: Real-time data compression library. *http://www.oberhumer.com/opensource/lzo/ (2012)*

[13] NVIDIA: The CUDA Toolkit. *http://developer.nvidia.com/cuda-downloads*

[14] Chih-Chung, C. and Chih-Jen L.: libSVM – A library for Support Vector Machines *(2011)*

[15] Nissen, S.: Fast Artificial Neural Network Library (FANN). *http://leenissen.dk/fann/wp/ (2011)*

[16] Pecina, P., Toral, A., Way, A., Papavassiliou, V., Prokopidis, P. and Giagkou, M.: Towards Using Web-Crawled Data for Domain Adaptation in Statistical Machine Translation. *Proceedings of the 15th Annual Conference of the European Association for Machine Translation*, pages 297-304, Leuven, Belgium, 2011.

[17] Bajernee et al.: METEOR – Automatic MT Evaluation Metric. *CMU Language Technologies Institute.*

[18] UFAL CUNI: Morce. http://ufal.mff.cuni.cz/morce/license.php *(2011)*

[19] Varga, D., Németh, L., Halácsy, P., Kornai, A., Trón, V., Nagy, V.: Parallel corpora for medium density languages. *In Proceedings of the RANLP (2005)*

[20] BMČ: Bibliographia medica Čechoslovaca. *2011*

[21] Dědek, J.: Report on and prototype of multilingual information extraction (Khresmoi Deliverable 4.1.2). *Khresmoi, 2012*

[22] Seznam: http://www.seznam.cz/ *Public website.*

[23] Centrum: http://www.centrum.cz/ *Public website.*

[24] Hladká, Hajič et al.: The Prague Dependency Treebank 2.0. *(2006)*

[25] Manevitz, L., Yousef, M.: One-Class SVMs for Document Classification. *The Journal of Machine Learning Research (2002)*

[26] Alpaydin, E.: Introduction to Machine Learning. *The MIT Press (2009)*

[27] Och, F., Ney, H.: A Systematic Comparison of Various Statistical Alignment Models. *(GIZA++, http://code.google.com/p/giza-pp/), (2003)*

[28] Papineni, K. et al.: BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, (2002)*

9 Appendix

9.1 Multilingual API

CUNI will provide SMT services for translation between several language pairs. As described above we have already trained Czech to English (and partially English to Czech models). The API will be common for all language pairs and can be summarized as follows:

- REST/JSON protocol – simple text protocol over HTTP, both requests and responses are JSON encoded documents
- Support for compression
- Support for security – searched information should be considered as highly personal
- Support for batch and asynchronous request (not considered in Year 2)

Client sends the JSON encoded query in the HTTP POST query to one of the CUNI servers. Server then responds with translations. The user can specify the language pair, type of translation and many additional parameters as how many translations she demands. Translations are sorted by the decreasing score. Because SMT is computationally very expensive and translation can take some time user can specify the tradeoff between quality and speed (latency). Latency can be decreased in SMT e.g. by phrase-table pruning. On the other hand, this obviously possibly decreases the quality of the translation.

```
POST /<cuni>/khresmoi/translate HTTP/1.1
```

```
{
   "sourceLang": "cs",
   "targetLang": "en",
   "docType": "document",
   "profile": "general public",
   "text": "Mám chřipku a bolí mě hlava. Také se potím.",
   "nbest": 1,
   "align-info": false,
   "async":                                                    false,
}
```

The response can look like this:

```
{
   "translation":
   [{
   "translated": "I have the flu and my head hurts. I also sweat.",
   "score": 768,
   "translationId":                "adeb8b91-c27f-4e95-a36c-62a7060e123b"
   }]
}
```

9.2 Czech Stopwords Used

aby, ako, akorát, ale, and, ani, ano, apod, asi, atd, během, bez, beze, blízko, bohudík, bohužel, bokem, buď, bude, budem, budeme, budeš, budete, budiž, budou, budu, bůhvíco, bůhvíčí, bůhvíjak, bůhvíjaký, bůhvíkam, bůhvíkde, bůhvíkdo, bůhvíkdy, bůhvíkolik, bůhvíkterý, bůhvínač, bůhvíproč, bych, bychom, byl, byla, byli, bylo, byly, bysme, být, cca, cokoli, cokoliv, copak, cosi, což, cože, častěji, často, čeho, čehokoli, čehokoliv, čehosi, čehož, čem, čemkoli, čemkoliv, čemsi, čemu, čemukoli, čemukoliv, čemusi, čemuž, čemž, čertvíco, čertvíčí, čertvíjak, čertvíjaký, čertvíkam, čertvíkde, čertvíkdo, čertvíkdy, čertvíkolik, čertvíkterý, čertvínač, čertvíproč, číhokoli, číhosi, číchkoli, číchsi, číkoli, čím, čímakoli, čímasi, čímikoli, čímisi, čímkoli, čímkoliv, čímpak, čímsi, čímukoli, čímusi, čímž, čísi, dál, dále, daleko, další, dám, dle, dnem, dnes, dneska, dobrá, dobré, dobrý, dobře, docela, dokonce, doposavad, doposud, doprostřed, dosavad, dospod, dospodu, dost, dosti, dosud, dovnitř, eště, formou, ho, hodinou, hodně, horší, hůř, hůře, chceš, chci, chtěl, jacíkoli, jacíkoliv, jacípak, jacísi, jak, jakákoli, jakákoliv, jakápak, jakási, jaké, jakéhokoli, jakéhokoliv, jakéhopak, jakéhosi, jakékoli, jakékoliv, jakémkoli, jakémkoliv, jakémpak, jakémsi, jakémukoli, jakémukoliv, jakémupak, jakémusi, jaképak, jakési, jakmile, jako, jakou, jakoukoli, jakoukoliv, jakoupak, jakousi, jakož, jakpak, jaký, jakýchkoli, jakýchkoliv, jakýchpak, jakýchsi, jakýkoli, jakýkoliv, jakýmakoli, jakýmakoliv, jakýmapak, jakýmasi, jakýmikoli, jakýmikoliv, jakýmipak, jakýmisi, jakýmkoli, jakýmkoliv, jakýmpak, jakýmsi, jakýpak, jakýsi, jakže, jasné, jasně, jde, je, jediná, jediné, jediný, jeho, jehož, jej, její, jejíhož, jejich, jejichž, jejíchž, jejímaž, jejímiž, jejímuž, jejímž, jejíž, jejž, jelikož, jemu, jemuž, jen, jenom, jenž, jenže, jestli, ještě, jež, ježto, ji, jí, jich, jichž, jim, jím, jimi, jimiž, jimž, jímž, jiná, jinak, jiné, jinou, jiný, jiných, jiným, jisté, jistě, již, jíž, jménem, jsem, jseš, jsi, jsme, jsou, jste, kam, každý, kde, kdeco, kdečí, kdejaký, kdekdo, kdekterý, kdepak, kdesi, kdo, kdokoli, kdokoliv, kdopak, kdosi, kdovíjak, kdovíkde, kdovíkdo, kdož, kdy, kdysi, když, kohokoli, kohokoliv, kohopak, kohosi, kohož, kol, kolem, kolik, kolikže, kolkolem, komkoli, komkoliv, kompak, komsi, komu, komukoli, komukoliv, komupak, komusi, komuž, komž, koncem, konče, končí, končíc, konec, kontra, kromě, která, kterákoli, kterákoliv, kterási, kterážto, které, kteréhokoli, kteréhokoliv, kteréhosi, kteréhož, kterékoli, kterékoliv, kterém, kterémkoli, kterémkoliv, kterémsi, kterémukoli, kterémukoliv, kterémusi, kterémuž, kterémžto, kterési, kteréžto, kterou, kteroukoli, kteroukoliv, kterousi, kteroužto, který, kterýchkoli, kterýchkoliv, kterýchsi, kterýchžto, kterýkoli, kterýkoliv, kterým, kterýmakoli, kterýmakoliv, kterýmasi, kterýmikoli, kterýmikoliv, kterýmisi, kterýmiž, kterýmkoli, kterýmkoliv, kterýmsi, kterýmžto, kterýsi, kterýžto, kteří, kteřísi, kteřížto, kříkoli, kříkoliv, kupodivu, kupříkladu, kvůli, kýmkoli, kýmkoliv, kýmpak, kýmsi, kýmž, lecco, leccos, lecčems, lecjak, lecjaký, leckam, leckams, leckde, leckdo, leckdy, leckterý, ledaco, ledacos, ledačí, ledajak, ledajaký, ledakdo, ledakterý, ledaskam, ledaskde, ledaskdo, ledaskdy, lépe, lepší, líp, má, mají, málo, máloco, málokdo, málokterý, mám, máme, máš, máte, max, mé, mě, mého, měl, měla, mělo, mém, mému, mezi, mi, mí, mimo, min, míň, místo, mít, mne, mně, mnoho, mnou, moc, mohl, mohla, mohou, mohu, moje, moji, mojí, mou, možná, mu, můj, musel, muset, musí, musím, musíš, musíte, může, můžeš, můžete, můžu, my, mých, mým, mými, nač, načež, načpak, nad, nade, nám, námi, namísto, naň, naprosto, naproti, např, napříč, nás, náš, naši, navíc, navrch, navrchu, navzdory, ně, nebo, nebude, nebyl, nebyli, nebyly, něco, něčí, nedaleko, nehledíc, něho, něhož, nechceš, nechci, nechť, nechtěl, něj, nějak, nějaká, nějaké,

nějakého, nějakou, nějaký, nejasné, nejasný, nejčastěji, nejde, nejen, nejhůř, nejhůře, nejlépe, nejnižší, nejsem, nejsou, nejvyšší, nějž, někam, někde, někdo, někdy, několik, nekončí, některý, nelze, něm, nemá, nemají, nemálo, nemám, nemáme, nemáš, nemáte, nemít, nemohl, nemohla, nemohou, nemohu, němu, nemusel, nemuset, nemusí, nemusím, nemusíš, němuž, nemůže, nemůžeš, nemůžete, nemůžu, němž, není, nepřesná, nepřesné, nepřesně, nepřesný, nepřímo, netřeba, netuším, netýká, neví, nevím, nevíš, nevlastní, nevyjímaje, nevyjímajíc, než, něž, ni, ní, nic, ničeho, ničem, ničemu, ničí, ničím, nie, nieje, nich, nichž, nijaký, nikdo, nikto, nim, ním, nimi, nimiž, nimž, nímž, nízká, niž, níž, nižádný, níže, nižší, nový, nutně, oba, obě, oběma, obou, oč, očpak, ode, odspoda, odspodu, ohledně, okamžikem, okolo, on, oň, ona, onen, oni, ono, ony, opravdu, oproti, ostatní, osum, pak, poblíž, počátkem, počínaje, počínajíc, pod, pode, podél, podle, podobně, pokud, poměrně, pomocí, ponad, pořád, poslední, posléze, posud, potom, pražádný, pro, proč, pročpak, proň, prostě, proti, proto, protože, před, přede, předem, přes, přese, přesná, přesné, přesně, přesný, při, přičemž, přímo, případná, případné, případně, případný, přitom, půlí, raději, rokem, sám, sama, samá, samé, samého, samém, samému, sami, samo, samou, samozřejmě, samozřejmý, samu, samy, samý, samých, samým, samými, se, sebe, sebou, sem, ses, si, sice, sis, skoro, skrz, skrze, snad, sobě, som, sotva, sotvaco, sotvakdo, spíš, spíše, spodem, spolu, stačí, stejně, stranou, středem, svá, své, svého, svém, svému, sví, svoje, svoji, svojí, svou, svrchu, svůj, svých, svým, svými, špatná, špatné, špatně, špatný, tací, tady, tahle, tak, taká, také, takhle, takováto, takové, takovéhoto, takovémto, takovémuto, takovéto, takovíto, takovouto, takový, takovýchto, takovýma, takovýmato, takovýmito, takovýmto, takovýto, takto, taky, taký, takže, tam, tamten, tatáž, tato, táž, tě, tebe, tebou, teď, teda, tedy, téhle, téhož, těchhle, těchto, těchže, těm, téma, těmahle, těmhle, těmihle, těmito, těmto, těmu, témuž, témž, témže, ten, tenhle, tenhleten, tento, tentýž, této, téže, ti, tihle, tím, tímhle, tímtéž, tímto, titíž, tito, tíž, tobě, tohle, toho, tohohle, tohoto, tom, tomhle, tomtéž, tomto, tomu, tomuhle, tomuto, totéž, toto, touhle, toutéž, touto, touž, touže, trochu, trošku, třeba, tuhle, tutéž, tuto, tvá, tvé, tvého, tvém, tvému, tví, tvoje, tvoji, tvojí, tvou, tvůj, tvých, tvým, tvými, ty, tyhle, týchž, týká, týmiž, týmž, tys, tytéž, tyto, týž, úderem, uplná, uplné, úplně, úplný, uprostřed, určitě, uvnitř, úvodem, vám, vámi, vás, váš, vaše, vaši, včetně, vedle, velmi, veprostřed, versus, vespod, vespodu, veškerý, vevnitř, víc, více, vím, vinou, víš, viz, vlastně, vlivem, vně, vnitřka, vnitřkem, vnitřku, von, vrchem, však, vše, všecek, všecka, všecko, všecky, všeho, všech, všechen, všechna, všechno, všechnu, všechny, všelico, všelicos, všeličehos, všeličems, všeličemus, všeličí, všeličíms, všelijaký, všelikdo, všeliký, všeliskdo, všem, všemi, všemu, vši, vší, všicci, všichni, vším, vůbec, vůči, vy, vyjma, vysoká, výše, vyšší, vzdor, vzhledem, vždy, za, zač, začátkem, začpak, zaň, zásluhou, zatím, závěrem, zboku, zcela, zčásti, zda, zdaleka, zde, zespoda, zespodu, zevnitř, zeza, znovu, zpět, zpod, zponad, zpoza, zprostřed, zřídkaco, zřídkakdo, zvnitřka, zvnitřku, žádný