VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

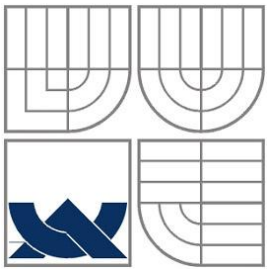
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

CZECH-ENGLISH TRANSLATION

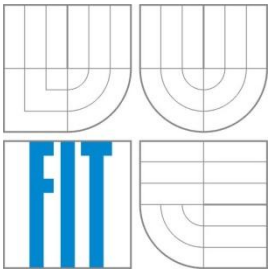
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JIŘÍ PETRŽELKA
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

CZECH-ENGLISH TRANSLATION
PŘEKLAD Z ČEŠTINY DO ANGLIČTINY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JIŘÍ PETRŽELKA
AUTHOR

VEDOUCÍ PRÁCE — doc. RNDr. PAVEL SMRŽ, Ph.D.
SUPERVISOR

BRNO 2010

Brno University of Technology - Faculty of Information Technology

Department of Computer Graphics and Multimedia — Academic year 2009/2010

Master Thesis Specification

For: **Petrželka Jiří, Bc.**

Branch of study: Intelligent Systems

Title: **Czech-English Translation**

Category: Artificial Intelligence

Instructions for project work:

1. Get acquainted with the existing statistical machine translation methods and the existing NLP tools for Czech and English.
2. Prepare parallel corpus for the standard evaluation of the developed system.
3. Design and implement a system able to translate free text in Czech into English, based on the collected parallel data.
4. Evaluate the realized system by means of standard metrics and compare it to alternative solutions.

Basic references:

- Manning, C. D., Schütze, H., Foundations of Statistical Natural Language Processing, MIT Press, 1999, ISBN 0-262-13360-1.

The Term Project discussion items:

- funkční prototyp řešení

Detailed formal specifications can be found at http://www.fit.vutbr.cz/info/szz/

The Master Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Smrž Pavel, doc. RNDr., Ph.D.**, DCGM FIT BUT

Beginning of work: September 21, 2009

Date of delivery: May 26, 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

Jan Černocký
Associate Professor and Head of Department

Abstract

This Master's thesis describes the principles of statistical machine translation and demonstrates how to assemble the Moses statistical machine translation system. In the preparation step, a research on freely available bilingual Czech-English corpora is done. An empirical analysis of time requirements of multithreaded word alignment tools demonstrates that MGIZA++ can achieve a five-fold speed-up, while PGIZA++ can reach an eight-fold speed-up (compared to GIZA++).

Three scenarios of morphological pre-processing of Czech training data are tested, using simple unfactored models. While pure lemmatization can aggravate the BLEU, more sophisticated approaches usually raise BLEU. The positive effect of morphological pre-processing diminishes as corpus size rises. The relation between other corpora characteristics (size, genre, extra data) and the resulting BLEU are empirically gauged. A final system is trained on the CzEng 0.9 corpus and evaluated on the testing set from WMT 2010 workshop.

Tato diplomová práce popisuje principy statistického strojového překladu a demonstruje, jak sestavit systém pro statistický strojový překlad Moses. V přípravné fázi jsou prozkoumány volně dostupné bilingvní česko-anglické korpusy. Empirická analýza časové náročnosti vícevláknových nástrojů pro zarovnání slov demonstruje, že MGIZA++ může dosáhnout až pětinásobného zrychlení, zatímco PGIZA++ až osminásobného zrychlení (v porovnání s GIZA++).

Jsou otestovány tři způsoby morfologického pre-processingu českých trénovacích dat za použití jednoduchých nefaktorových modelů. Zatímco jednoduchá lemmatizace může snížit BLEU, sofistikovanější přístupy většinou BLEU zvyšují. Positivní efekty morfologického pre-processingu se vytrácejí s růstem velikosti korpusu. Vztah mezi dalšími charakteristikami korpusu (velikost, žánr, další data) a výsledným BLEU je empiricky měřen. Koncový systém je natrénován na korpusu CzEng 0.9 a vyhodnocen na testovacím vzorku z workshopu WMT 2010.

Keywords

statistical machine translation, natural language processing, translation model, language model, decoder, word alignment, GIZA++, MGIZA++, PGIZA++, SRILM, hunalign, plain2snt, snt2cooc, mkcls, BLEU, bilingual corpus, Kačenka, Acquis Communautaire, CzEng, OpenSubtitles, hidden Markov model, HMM, viterbi, IBM model, Qin Gao, ÚFAL, IFAL, EuroMatrix, Moses, Czech morphology, lemmatization, Prague Dependency Treebank, PDT, Libma, BLEU, WMT

statistický strojový překlad, zpracování přirozeného jazyka, překladový model, jazykový model, dekodér, zarovnání slov, GIZA++, MGIZA++, PGIZA++, SRILM, hunalign, plain2snt, snt2cooc, mkcls, BLEU, bilingvní korpus, Kačenka, Acquis Communautaire, CzEng, OpenSubtitles, skrytý Markovův model, HMM, viterbi, IBM model, Qin Gao, ÚFAL, EuroMatrix, Moses, česká morfologie, lemmatizace, Pražský závislostní korpus, PDT, Libma, BLEU, WMT

Citation

Petrželka, J. *Czech-English Translation. Master's Thesis.* Brno, Brno University of Technology, 2010.

Declaration

The work described in this report is the result of my own investigations. All sections of the text and results that have been obtained from other work are fully referenced.

Signed: .....

26 May 2010

Acknowledgements and dedications

I would like to thank all those who supported me in the course of work on this thesis, which primarily includes my family.

I also thank my supervisor Pavel Smrž for expert consultation on the topic discussed in this work.

Table of contents

Preface

Machine translation (MT) aims at substituting a human translator by a computer. In broader perspective, machine translation is a specific application of a scientific discipline called natural language processing (NLP). NLP is a computer science field. Apart from computer science, language processing derives insights from fields such as electrical engineering, linguistics and psychology (Jurafsky et al., 2009:9).

So far, quality translations from one language to another have not been common, except for the most restricted domains, such as weather reports (Manning, 1999:463). In most cases, it is necessary for the human translator to post-edit the output of MT. According to certain experiments (Plitt, 2010), this can considerably increase translators' productivity.

In the **European Union**, there is a growing need for high quality machine translation systems because the number of language combinations used in the EU rises with the entry of every new country. To address this issue, the EuroMatrix Project (2010a) and the The EuroMatrixPlus Project (EuroMatrixPlus Consortium, 2010) have been founded.

The **objective** of the **master's thesis** is to design, implement and evaluate a statistical machine translation system.

In chapter 1, we start by introducing some theoretical aspects of the machine translation. We do not intend to give a comprehensive account of all the mathematical aspects of machine translation. The objective here is to give the reader a basic idea of the process of building the entire machine translation system.

Following the theoretical introduction, **chapter 2** provides a step-by-step guide on the process of building a statistical machine translation (SMT) system. First, we research available Czech-English corpora and prepare them for use in our system. Next, we empirically analyze the benefits of multithreading when doing word alignment with the MGIZA++ and PGIZA++ alignment tools. Based on our observations, we create a SMT system based on the Moses SMT system. We then draw up a methodology to empirically assess the system's performance under several scenarios.

In chapter 3, we carry out the previously proposed experiments. We attempt to establish a relation between various corpora characteristics and the resulting BLEU score. We investigate how the size and genre of the corpus influence the BLEU. Next, we attempt to raise the BLEU by introducing additional information into the system. We include a dictionary into the training

data and we use an extra corpus to train the language model. Subsequently, we analyze the system's performance under three different schemes of morphological pre-processing. Finally, we train the system on the CzEng 0.9 corpus and evaluate it on the testing data from the WMT 10 workshop.

This Master's thesis draws on the **Term project**. The core of the first chapter (sections 1.1, and 1.2.1 to 1.2.4) from the Term project has been used and supplemented with additional information on morphology (section 1.2.5). In chapter 2, information on corpora (section 2.1) has been updated and extended. Section 2.2 has been borrowed from the term project almost without change. The following sections (starting with section 2.3) and the entire chapter 3 are novel additions first appearing in this thesis.

1 Theoretical approaches to machine translation

We can conceptually divide the strategies for MT into two categories: **classical MT** and **statistical MT**. Practical applications nowadays combine these two approaches so think of the concepts introduced in this chapter as ideas that can be incorporated to various extents in MT systems.

1.1 Classical MT

The most trivial idea of how we could translate a text from one language to another would probably be: Take the words from the source text, one by one, and by means of a dictionary, substitute them with corresponding words from the target language. This approach is called **direct translation**. Before substituting the words we usually need to do some morphological analysis on the source text.

Although direct translation is usually not feasible for distant language pairs, it can be used for close language pairs, such as Czech-Polish or Czech-Lithuanian where the syntactic constructions of both languages are almost identical. This approach has been used in the **Česílko** system (Cuřín et al., 2007).

A more sophisticated approach is to analyze the source language text syntactic structures. Once we have got a parse tree of the source text, we transform the tree so that it conforms to syntactic structures of the target language[1]. We can also make use of semantic information. These two approaches are generally called **transfer** approaches. In practise, this architecture has been used in the **Dependency-based Machine Translation system** developed at ÚFAL (Cuřín et al., 2007).

To see the context of the options discussed so far, please refer to Figure 1.1, depicting the Vauquois triangle, which shows the individual levels at which the language can be analyzed. So far, we have described all the "floors" except for the one at the top.

[1] This is done using *contrastive knowledge* – e.g. knowing that adjectives in the source language come before nouns but in the target language, they come after nouns etc.

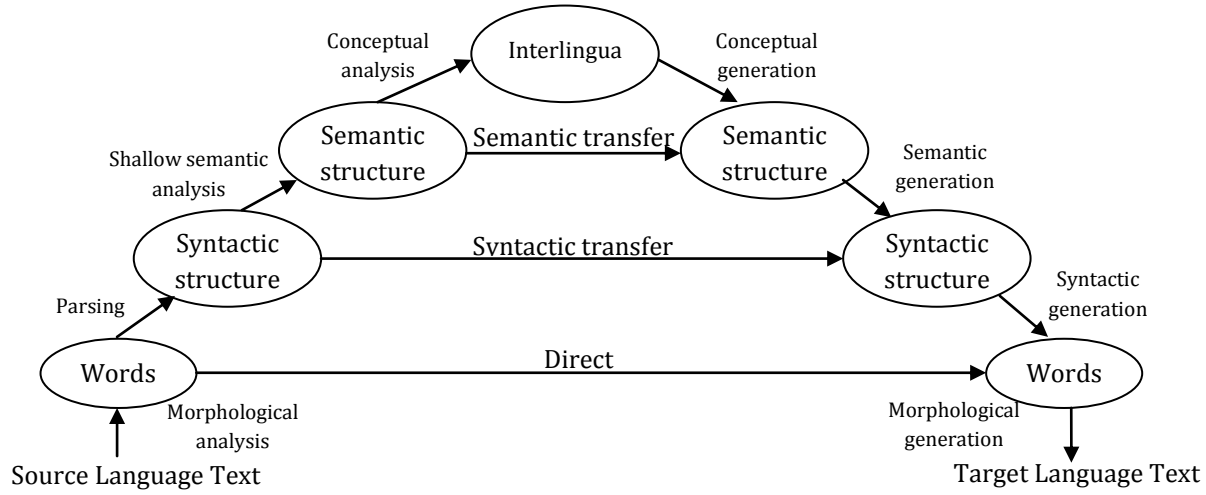


Figure 1.1 Vauquois triangle

At the top of the Vauquois triangle, there is the **interlingua** approach. In this case, we analyze the source language text and save it into an abstract representation called interlingua. Then we can generate the target text directly from the interlingua. The advantage of this approach is that we can use the interlingua representation to generate the target text in any language. However, there are other problems to tackle[2].

1.2 Statistical MT

Statistical MT differs from the classic architectures in that it concentrates on the result, not the process of translating. What we want is a translation that reads fluently and is faithful in respect to the original sentence. Jurafsky et al. (2009:875) exemplifies this by the Hebrew *adonai roi* (*"The lord is my shepherd.")* that cannot be literally translated into a language that has no sheep. We can either say something like *"the Lord will look after me"* or *"the Lord is for me like somebody who looks after animals with cotton-like hair".* The first translation is clear in the target language but is only partially faithful to the original. The second translation, on the other hand, is faithful to the original but reads awkwardly in the target language. The task of a human translator is to compromise between fluency and faithfulness and this is exactly what statistical MT systems attempt to do as well.

We can formalize the idea as follows (T denoting target, S denoting source):

$$\text{best translation } \hat{T} = argmax_T(faithfulness(T,S)fluency(T)) \tag{1.1}$$

We choose such a target language sentence that has the maximum product of faithfulness and fluency.

[2] For example, if the interlingua distinguishes between elder brother and younger brother (which is necessary for Japanese and Chinese) then it will have to compute a lot of unnecessary disambiguation when translating between English and Czech where there is only one concept for a brother.

To further formalize the idea, let us assume we translate a foreign language sentence $F = f_1, f_2, \dots, f_m$ to English. We are looking for the best English sentence $\hat{E} = e_1, e_2, \dots e_l$ whose probability $P(E|F)$ is the highest. Using the Bayes' rule we can rewrite this as follows:

$$\hat{E} = argmax_E P(E|F)$$

$$= argmax_E \frac{P(F|E)P(E)}{P(F)}$$

$$= argmax_E P(F|E)P(E) \tag{1.2}$$

The denominator $P(F)$ can be ignored because it is a constant. The resulting equation consists of two components – a **translation model** $P(F|E)$ and a **language model** $P(E)$. The last thing we need is a **decoder** which will be given *F* and it should produce *E*.

1.2.1 Language model

We need a description of the rules that govern the language we want to translate to. This description is called a language model (LM) and in statistical MT language models are based on *N*-grams. What are they?

Suppose you have to guess the next word in the sentence *Have a nice ...* You would agree that these three words will probably be followed by *day* or *weekend* but it is much less likely they will be followed by *at* or *nice*.

The idea of *N*-grams is exactly the same. More formally, given a sequence of words of length *N*-1, the model tries to predict what the *Nth* word will be. A 2-gram model is commonly called a **bigram** model, a 3-gram model is called a **trigram** model, a 4-gram is called a quadrigram (or tetragram) model etc. When we just say *N*-gram we either mean a word sequence of length *N* or a language model based on *N*-grams.

To create a language model we need a monolingual corpus of the target language and a toolkit for building a language model, for example the SRILM toolkit (SRI International, 2009).

1.2.2 Translation model

The translation model tells us the probability that a given English sequence of words *E* generates a foreign sequence of words *F*. In the case these sequences have a length of 1, we work with individual words (this is called **word-based** statistical MT) but in this thesis, we concentrate on entire chunks of words, called phrases (this is called **phrase-based** statistical MT).

How do we go about building a translation model?

First, we group the English sentence into phrases $\overline{e_1}, \overline{e_2}, \dots \overline{e_I}$. Next, we need to translate these phrases one by one into foreign phrases $\overline{f_i}$ and then to reorder the foreign phrases.

How do we enumerate the probability $P(F|E)$? It will rely on two factors – the **translation probability** (how likely is the given translation?) and **distortion probability** (how likely is a given reordering of phrases?). We will denote the probability of an English phrase being translated to a foreign phrase as $\phi(\overline{f_i}|\overline{e_i})$.

Next, we denote the distortion probability as *d*. The distortion probability means the probability of two consecutive English phrases being separated in the translation into a foreign language by a span of words of a particular length. Formally, $d(a_i - b_{i-1})$ denotes the distortion probability where a_i is the start position of the foreign phrase generated by the *i*th English phrase $\overline{e_i}$, and b_{i-1} is the end position of a foreign phrase generated by the (*i*-1)th English phrase $\overline{e_{i-1}}$. We can compute a simple distortion probability using the following formula: $d(a_i - b_{i-1}) = \alpha^{|a_i - b_{i-1} - 1|}$. In this way, we penalize the probability of a translation where the phrases lie far apart.

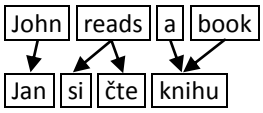
The final translation model for phrase-based MT is:

$$P(F|E) = \prod_{i=1}^{I} \phi(\overline{f_i}|\overline{e_i})\, d(a_i - b_{i-1}) \tag{1.3}$$

Now what we need is a list of English and foreign phrases and a probability they match together (the so called **phrase-translation table**). To create such a table manually would be too time consuming. Therefore we try to automate the process. First, we need parallel corpus on input. Then, for each sentence pair, we do a **word alignment** (we figure out which word in the English sentence corresponds to which word in the foreign sentence). Having the word alignment, we can extract phrases and produce the **phrase alignment** and the phrase-translation table.

1.2.2.1 Word alignment

What we want to achieve is a mapping between words in a source language sentence and words in a target language sentence, for example:



	Jan	si	čte	knihu
John				
reads				
a				
book				

Figure 1.2 Example of a simple word alignment

Notice that we allow here that one English word is mapped to any number of Czech words and a Czech word can be mapped to any number of English words.

There exist several algorithms for word alignment. They differ in the level of sophistication. The most popular are the IBM models 1, 3, 4, 5 and the HMM model (HMM being a better alternative to the IBM model 2). However, these models align the words under the assumption that the

mappings can only be one-to-many (one word from the source language aligns to one or more words in the target language).

We should note that we usually add a fictitious NULL word in the source sentence which can map to a word in the target sentence that has no real equivalent in the source sentence.

When training the model, we need a parallel corpus. A parallel corpus is a text that is available in two languages. More formally, we need a corpus consisting of S sentence pairs $\{(F_s, E_s)|s = 1 \dots S\}$. We use this corpus as input to a tool that can align words. A standard for word alignment is currently the GIZA++ tool (Och, 2001) which is based on the IBM and HMM models mentioned above.

How do we tackle the problem with the restricting one-to-many assumption? We simply do the word alignment in both directions (English ⟶ Czech, Czech ⟶ English) and then do an intersection (or other sensible operation) of the two matrices.

Once we have the word alignment matrix, we compute the phrase-translation table.

1.2.3 Decoder

The decoder first takes the original sentence and divides it into phrases. (If we were doing a word-based MT it would be words, not phrases.) There are usually many ways how to divide a sentence into phrases. They are called **translation options** (Figure 1.3 illustrates this).

John	reads	a	book
Jan	čte	nějakou	kniha
	si čte	knihu	
	si čte	knihy	
	čte knihu		

Figure 1.3 Translation options

Now the decoder starts generating the output sentence from left to right in the form of hypotheses (Figure 1.4), starting with an initial hypothesis. Then it expands it so that the phrase *John* is translated as *Jan*. We use an asterisk to denote that the first word has already been translated. Also, we record the probability of this translation (0.487). We can then decide to expand the tree further, which can yield *Jan si čte* with the probability 0.176.

Decoders are usually based on a best-first search algorithm (Jurafsky et al., 2009:890). This is an informed search that expands a node *n* based on the evaluation function *f(n)*.

The evaluation function in our case for partially translated phrases $S = (F, E)$ is based on the following formula:

$$cost(E, F) = \prod_{i \in S} \phi(\bar{f}_i, \bar{e}_i) d(a_i - b_{i-1}) P(E) \tag{1.4}$$

It is a product of the translation, distortion, and language model probabilities for all phrases that have been translated so far. This cost is usually called the **current cost**. It is usually combined with the estimated **future cost** because otherwise the algorithm would tend to select such translations that have a few high probability words at the beginning at the expense of translations with higher overall probability (Jurafsky et al., 2009:892).

Practical decoders like Moses (Euromatrix Project, 2008a) must prune the search space because the number of hypotheses would grow exponentially, so for example Moses uses a beam search algorithm rather than best-first search.

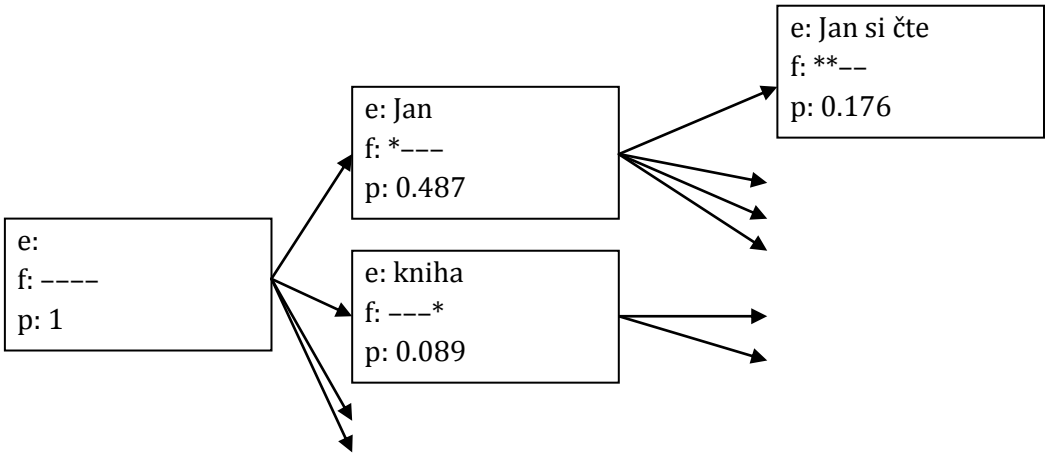


Figure 1.4 Generation of hypotheses

1.2.4 Evaluation

We need to measure the quality of the output produced by the MT system. There are generally two ways to do this – either a human evaluator can read the output sentences one by one and judge its fidelity and fluency, or we can evaluate the output using an automated program.

1.2.4.1 Human evaluation

Human evaluation can proceed in several different ways. The first way is to present the evaluator the output sentences and ask him to grade it on a scale, such us **fluency** or **fidelity**. Another way is to hide some words in the output sentence and ask the evaluator to fill in the missing word. In this case we measure the time it takes for the evaluator to fill in the word. This method is called a **cloze** task. Finally, we can give the evaluator the output sentences and ask them to post-edit the output so that it reads fluently. In this case we measure the **edit cost**, which can be the number of words needed to be replaced or the total edit time.

1.2.4.2 Automatic evaluation

Human evaluation can be costly and time consuming. Therefore we need an automated means of evaluating the output of the MT. The fundamental idea is to measure how similar the MT output is to a human translation. We can then easily run the evaluation on similar versions of a MT system and find out which one is better.

There are a number of heuristic methods which do this, such as **BLEU**, **NIST**, **TER**, **Precision and Recall**, and **Meteor** (Jurafsky et al., 2009:895). One of the most popular metrics nowadays is the BLEU (Bilingual Evaluation Understudy).

The BLEU takes a MT output sentence and computes the weighted average of the number of *N*-grams overlapping with the corresponding human translation:

$$p_n = \frac{\sum_{C \in \{Candidates\ \}} \sum_{n-gram \in C} \text{Count(n-gram)}}{\sum_{C' \in \{Candidates\ \}} \sum_{n-gram' \in C'} \text{Count(n-gram')}} \tag{1.5}$$

BLEU uses unigrams, bigrams, trigrams and quadrigrams and combines these precisions by taking their geometric mean (Jurafsky et al., 2009:897).

BLEU is generally a good choice when evaluating several versions of the same MT architecture. However, it performs poorly when cross evaluating different architectures. It also focuses too much on local information and may therefore rank higher than a human evaluator would.

1.2.5 Morphology

A specific issue we will address in this thesis is morphology. The motivation follows from the fact that Czech is a morphologically rich language while English is not[3].

Morphology studies the way words are built from smaller units called **morphemes**. For example, the word *cars* consists of two morphemes – *car* and *s*. Morphemes can be divided into two classes – stems and affixes (stem being *car* in the previous example; *-s* being an affix). We can further subdivide affixes into prefixes, suffixes, infixes and circumflexes. Prefixes precede the stem, suffixes follow the stem, circumflexes do both, and infixes are inserted inside the stem (Jurafsky et al., 2009:47).

Morphemes can be combined to create new words. This can happen through inflection, derivation, compounding and cliticization. **Inflection** is the combination of a word stem with a morpheme usually resulting in a word of the same class. The new word usually adds some syntactic information, for example the word *cars* is created by inflection from the word *car*. The *–s* suffix tells us it is plural. **Derivation** is a combination of a word stem and a morpheme, resulting in a word from a different class. For instance, the noun *binarization* is derived from the adjective *binary*. **Compounding** is a combination of multiple word stems together (e.g. *doghouse*). **Cliticization** is a combination of a word stem with a clitic (e.g. *I've* – the –'ve part is a clitic).

In this work we are going to address the inflectional morphology. One Czech word can occur in a corpus in many forms (for example, the English word *bowl* corresponds to the Czech forms *miska, misky, misce, misku, misko, miskou*). Unless we provide some additional information to the

[3] More specifically, we speak about inflectional morphology here.

system, the individual Czech word forms are treated separately, which can lead to **data sparseness** on the side of the Czech corpus.

Still another problem are **ambiguous words**, which are written identically but have different meaning. For instance, the word form *mnou* is both a pronoun in instrumental case meaning *with me* and a plural verb in the third person meaning *they rub*.

One way to work with Czech inflectional morphology is to **ignore ambiguity** and data sparseness and give the system a very large corpus so that the probabilistic rules of the translation and language model infer these rules like any other rules.

Another approach is to use **factored models**. Instead of training only on the word factor (which we discussed earlier), we train on additional factors, e.g. on the word lemmas, word class, part-of-speech etc. Figure 1.5 illustrates this. Now, instead of the word *mnou,* the input corpus could, in a simple case, contain *mnou|já|pronoun.* The model will be trained on the lemma and word class as well and during decoding, a combination of these three factors will be evaluated.

A more simplistic approach is to use a model which is **not factored**. Goldwater et al. (2005) suggests several ways how to improve system performance. Apart from simple lemmatization or truncation of the Czech corpus they propose adding pseudo words to the Czech corpus that imitate the way English inflectional morphology works.

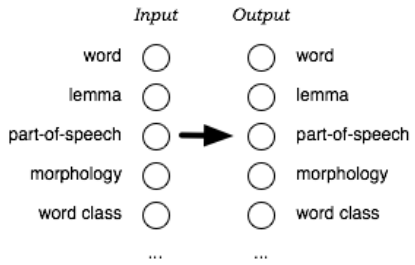


Figure 1.5 Vector of factors (Koehn, 2010)

1.2.5.1 Morphological annotation

A corpus can be annotated either manually or automatically. Manual annotation is time consuming and for large corpora not practical. Usually, the annotation is done automatically by a tool. The disadvantage of using an automatic annotator is that it may fail to analyze the word correctly.

There exist several tools that can analyze the input corpus and annotate each word with additional morphological information. In this thesis, we will be working with the **Prague**

Dependency Treebank 2.0 (Hana et al., 2005), more specifically with its m-layer, and with the **Libma library**[4] from Stanislav Černý.

To illustrate the output of the Prague Dependency Treebank, look at the morphological analysis of the word *hraniční*:

```
                        hraniční AAIS4----1A----
standard adjective, masculine inanimate, singular, accusative, positive.
```

There are 15 positions with clearly defined semantics. For example, the first position (A) indicates the word is an adjective.

The output of the Libma for the word *hraniční* is similar:

```
                         hraniční k2eAgNnSc5d1
adjective, affirmative, neutral, singular, vocative, positive.
```

The first two positions (k2) indicate that it is an adjective and so on.

Even this simple demonstration showed that automatic annotation may not be able to determine some characteristics unambiguously. We don't know if the word is in accusative or vocative and without context, we cannot even find out. The tool for automatic annotation may or may not take into account the neighbourhood of the word and disambiguate more or less correctly.

[4] Obtained from minerva1:/mnt/minerva1/nlp/local/share/Ma/libma and documented at https://merlin.fit.vutbr.cz/nlp-wiki/index.php/Morfologický_slovník_a_morfologický_analyzátor_pro_češtinu

2 Building a statistical machine translation system

2.1 Getting a parallel corpus

As indicated in chapter 1, statistical machine translation is based on unsupervised learning algorithms that need a large number of bilingual texts on input. Such a text is called a corpus (plural *corpora*). There are several possible sources of parallel Czech-English texts.

In this work, we will primarily work with these parallel Czech-English corpora: **Kačenka 2** (Šlancarová, 2003), **Acquis Communautaire** (European Commission, 2009), **OpenSubtitles** (Tiedermann, 2007), **CzEng 0.7** (Bojar et al., 2007), **CzEng 0.9** (Bojar et al., 2009a) and **WMT10** (described later).

2.1.1 Kačenka 2

The Kačenka 2 corpus has been created at the Faculty of Arts, Masaryk University and contains 16 bilingual Czech-English fiction books. Unfortunately, the available source[5] contains only paragraph-aligned plain texts so a preparation had to be done before we could use this corpus.

2.1.1.1 Preparation of the Kačenka 2 corpus[6]

First, all corpus plaintext files have been uniformly converted to utf-8. Next a kacenka.py script has been created and run on all these files. The script proceeds as follows: First it splits each book into separate English and Czech files. Then it erases all non-textual elements (such as <i>), leaving only the <p> element (which assists the hunalign, described below). Then it separates the paragraphs to sentences, writing each sentence on one line. Next, it cleans the file by erasing all quotation marks and omitting all sentences that are shorter than 2 or longer than 40 words. Subsequently, it tokenizes the files and runs the hunalign programme (Hunglish Project, 2009) to sentence-align the files. We use a bilingual Czech-English dictionary to help hunalign align the sentences. Then, we split the hunalign output into separate English and Czech files. Finally, we merge the outputs for each book and obtain two final sentence-aligned files: kacenka.en and kacenka.cs.

[5] Obtained from minerva1:/mnt/minerva1/nlp/corpora/parallel/KACHNA2_hotove_texty/HOTOVE&ALIGNED/.

[6] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/kacenka_preprocessing/

Apart from the hunalign tool, the kacenka.py script makes use of the clean_txt.py script[7], the merge.py script[8] and the tokenizer.perl script[9].

The original corpus contains 3 122 305 words. After preparation it contains 1 523 903 Czech tokens and 1 697 637English tokens (tokens being notably dots, commas and, of course, regular words). It contains **118 285 sentence pairs**.

2.1.2 Acquis Communautaire

The **Acquis Communautaire** (AC) corpus comprises of legislative texts of the European Union from the 1950s to now.

The corpus[10] contains **234 320 sentence pairs**, which is approximately double the size of Kacenka 2. However, the AC corpus' average sentence length is much greater than that of Kacenka. The number of Czech and English tokens in the AC is 5 804 785 and 6 752 251, respectively.

2.1.3 Open Subtitles

The **Open Subtitles** corpus consists of subtitles from movies. The corpus[11] contains **377 623 sentence pairs** but only 2 458 480 Czech and 3 086 874 English tokens. This is caused by the fact that the sentences are very short on average.

2.1.4 CzEng 0.7

This is the second largest corpus we will use. It has been compiled the ÚFAL (2007) and contains texts from multiple domains. Its sources are: Acquis Communautaire, Readers' Digest, Project Syndicate, KDE, GNOME, Kačenka, Navajo User Translations, E-Books, European Constitution and Samples from European Journal (Bojar, 2007).

We will use its pre-processed version[12]. The corpus consists of **1 096 940 sentence pairs** (15 292 171Czech and 17 868 659 English tokens).

2.1.5 CzEng 0.9

CzEng 0.9 is a new release of the CzEng corpus from ÚFAL. Similar to the CzEng 0.7, this corpus contains texts from various domains (movie subtitles, EU legislation, technical documentation, fiction, parallel web pages, news, project Navajo). However, its size about seven times bigger - it contains **8 029 801 sentence pairs**. The authors of this corpus say that their intent was to compose a large corpus, not a balanced corpus (Bojar et al., 2009b). They say that according to their findings, "larger datasets usually improve the quality of MT, even if the additional data are out of the translated domain" (Bojar et al., 2009b).

[7] Obtained from minerva1:/mnt/minerva1/nlp/projects/ac_books/tools/clean_txt/
[8] Obtained from minerva1:/mnt/minerva1/nlp/projects/ac_books/tools/others/
[9] Obtained from minerva1:/mnt/minerva1/nlp/projects/ac_books/tools/others/
[10] Obtained from minerva1:/mnt/minerva1/nlp/corpora/parallel/Acquis_Communautaire/xschmi01/
[11] Obtained from minerva1:/mnt/minerva1/nlp/projects/ac_books/corpora/opus/
[12] Obtained from minerva1:/mnt/minerva1/nlp/projects/ac_books/corpora/czeng/

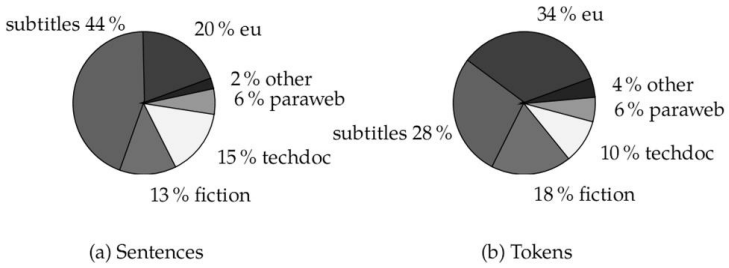


Figure 2.1 Proportions of texts in the CzEng 0.9 corpus (Bojar et al., 2009b)

2.1.5.1 Preparation of CzEng 0.9 corpus[13]

The CzEng 0.9 corpus is freely available for non-commercial purposes (Bojar et al., 2009b). There are three versions of the corpus – apart from the plaintext version there are two more versions that contain additional morphological and syntactic information. In this thesis we work only with the plaintext version.

Apart from extracting and merging all the parts of the CzEng 0.9 it is to note that we also did a specific cleaning to remove apostrophes and quotation marks from the corpus. The official statistics and our statistics of the corpus therefore slightly differ in the number of tokens.

2.1.6 WMT10

This is our working name for the training, development and test sets from the Translation task of the Fifth Workshop on statistical Machine Translation (European Commission, 2010a). The training data are a combination of about 45 million words from the Europarl corpus and about 2 million words from the News Commentary corpus (European Commission, 2010b).

2.1.7 Other corpora

In addition to the corpora presented so far, we will occasionally make use of other corpora. This section gives a brief summary.

First, there is the **Books 2** corpus[14], composed by Radek Bartoň (2010) from FIT, Brno University of Technology. This corpus has similar characteristics as the Kacenka corpus because it is composed exclusively by fiction books. The only difference is its size – it contains about 8 times more sentences than the Kacenka corpus.

Apart from ordinary corpora, we will also make use of Czech-English dictionaries[15] later. We will work with them in the very same way as with corpora. The **Lite Dict** corpus is a dictionary listing almost exclusively word-to-word records. The **Full Dict** corpus is a dictionary containing multiword phrases as well.

[13] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/czeng_preprocessing/
[14] Obtained from minerva1:/mnt/minerva1/nlp/projects/ac_books/corpora/books2/ and presented at https://merlin.fit.vutbr.cz/nlp-wiki/index.php/Parallel_Corpus_Joint-Multigram_Training_2
[15] Obtained from minerva1:/mnt/minerva1/nlp/projects/ac_books/tools/hunalign/dict/

2.1.8 Unused corpora[16]

Apart from the above mentioned corpora, there are other sources on the Minerva1 server.

The **ČNPK** (Czech-German parallel corpus, Pelouškova, 2007). This corpus is useless for building the translation model since we concentrate on Czech-English translation.

Similarly, the **PECT** directory cannot be utilized for building a translation model because, in fact, this directory contains a monolingual corpus consisting of extracts from the Lidové Noviny newspapers. Despite being a good source for building a language model if translating from English to Czech, it would be first necessary to pre-process the data.

The **terminologie** directory contains technical texts, mostly in the PDF format. These texts could possibly be used for building a translation model but the data would need to be pre-processed first. This task, however, would exceed the time quota allocated for this master thesis. The clean-up would not be trivial because the corpus also contains words from other languages than Czech and English (e.g. French). The terminology directory has therefore not been used.

2.2 Analysis of time requirements of word alignment tools

For word alignment, we will be using the **MGIZA++** and **PGIZA++** tools (Gao, 2009). They are based on the standard GIZA++ (Och, 2001). Both MGIZA++ and PGIZA++ have been developed with the idea in mind that the most of the alignment process can run in parallel. More specifically, the IBM and HMM alignment models used by these tools are an implementation of the EM algorithm (Dempster et al., 1977; In: Gao et al., 2008), which means that the algorithm runs for a number of iterations. In each iteration, the best word alignment for each sentence pair is first computed. Once all the alignments are known, the algorithm normalizes the counts and proceeds to next iteration. The important thing is that the word alignment, being the most time-consuming step, can run in parallel.

The MGIZA++ exploits this parallelism by using multithreading on a multiprocessor system. It spawns several processes which do the alignment in parallel, using a common address space and a mutual locking mechanism. The disadvantage of the MGIZA++ is its lack of scalability (the top being the maximum number of CPUs available).

The PGIZA++, on the other hand, runs on a cluster of autonomous computers. The corpus is split to parts and each node works on its part of the corpus. The machines communicate via the SSH remote procedure call. The advantage of PGIZA++ is its scalability, while its disadvantage is the need to transfer big amounts of data using the I/O.

[16] Found at minerva1:/mnt/minerva1/nlp/corpora/parallel/

2.2.1 MGIZA++

First, we had to compile the MGIZA++ application[17].

The MGIZA++ is run in the same way as the standard GIZA++, except that it supports the NCPUS argument which allows us to define the number of threads which will be used for training. If the NCPUS equals 1 then the MGIZA++ works like the standard GIZA++. Another thing we have to do after the training is to run a script to merge the aligned parts from individual threads together.

2.2.1.1 Training with MGIZA++

Before the actual training process can be started, we have to run three tools: *plain2snt*, *snt2cooc*, and *mkcls*.

The **plain2snt** tool takes the corpus file on input and produces two files – one with a *vcb* extension, which contains all the words from the corpus together with a unique number, and another file with a *snt* extension, which contains the original corpus with all the words replaced by their numerical indices specified in the vcb file. This is done to speed up the subsequent GIZA++ run (so that it can work with numbers, not with strings).

The **snt2cooc** tool creates a co-occurrence file.

The **mkcls** tool creates word classes. Running this tool took from several minutes (Kacenka corpus) to about an hour (CzEng 0.7 corpus).

Now we can run the **MGIZA++** tool. There are a number of parameters we can set at the GIZA start-up; a comprehensive list is available at (Gao, 2009b). We decided to leave the implicit parameters. All we want now is to see the potential speed-up when aligning various corpora using various numbers of threads. The motivation now is to find out how much time we can spare when using MGIZA++ over standard GIZA++. Based on this knowledge, we will later be able to quickly train various corpora with various ways of pre-processing and analyze the quality of the translation with BLEU.

The MGIZA++ with its implicit parameters trains five iterations of IBM model 1, five iterations of the HMM, five iterations of the IBM model 3, and five iterations of the IBM model 4 (the last two being denoted as the Viterbi model).

After the MGIZA++ run is completed, a script (merge_alignment.py) must be run to merge the alignments from individual threads. The script can be obtained from (Gao, 2009).

The entire training process has been automated with the traingiza.py script (see Appendix A – Scripts created).

[17] After several unsuccessful attempts we learnt that MGIZA++ cannot be compiled with GCC 4.3 or greater (Google Code, 2009). Therefore, the Makefiles had to be rewritten to use the GCC 4.1 compiler. After this step, the program could be successfully compiled.

Unless otherwise stated, the testing has been carried out on the athena3.fit.vutbr.cz server. Currently, this server contains 8 CPUs, each having the speed of 2.6 GHz.

2.2.1.2 MGIZA++ on the Kacenka 2 corpus

First, we wanted to see what the potential speed-up can be when word-aligning a small corpus. We ran MGIZA++ for the Kacenka 2 corpus for 1, 2, ..., 8 threads. Figure 2.2 illustrates the results.

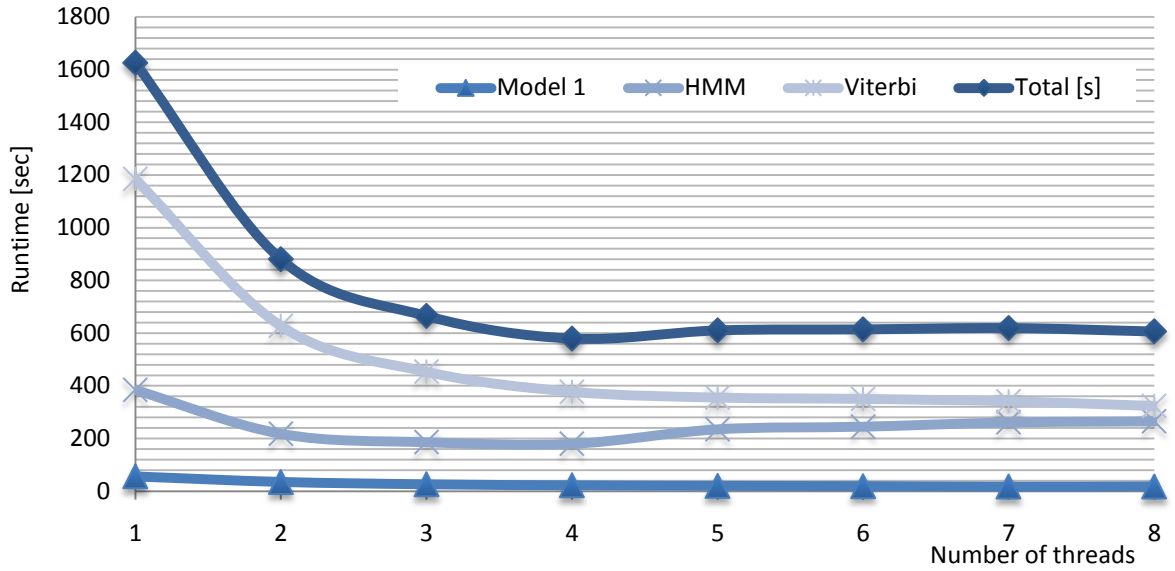


Figure 2.2 MGIZA++ run for the Kacenka 2 corpus on Athena3[18]

Model/threads	1	2	3	4	5	6	7	8
Model 1	56	35	26	22	20	19	17	18
HMM	384	218	186	180	235	245	261	265
Viterbi	1185	627	453	377	355	350	341	323
Total [s]	1625	880	665	579	610	614	619	606
Total [min]	27	15	11	10	10	10	10	10
Speed-up	100%	54%	41%	36%	38%	38%	38%	37%

Table 2.1 MGIZA++ run for the Kacenka 2 corpus on Athena3

As you can see, using up to 4 threads to parallelize the process yields almost a speed-up of 1:3. However, adding more threads is counterproductive. The reason for this is probably the mutual locking mechanism used to synchronize the threads (Gao et al., 2008).

The numbers suggest, however, that using more than 4 threads leads to a slight speed-up in the Viterbi training. Could this speed-up outweigh the increasing cost of the HMM training if we trained on a larger corpus?

[18] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/mgiza_tests/kacenka_ncpus[1-8]/

2.2.1.3 MGIZA++ on the CzEng 0.7 corpus

To find out if more that 4 threads are of any use when training on a large corpus, we repeated the training from previous section – this time, using the CzEng 0.7 corpus (see Figure 2.3). The testing for 7 CPUs has been omitted (it would probably bring no new information).

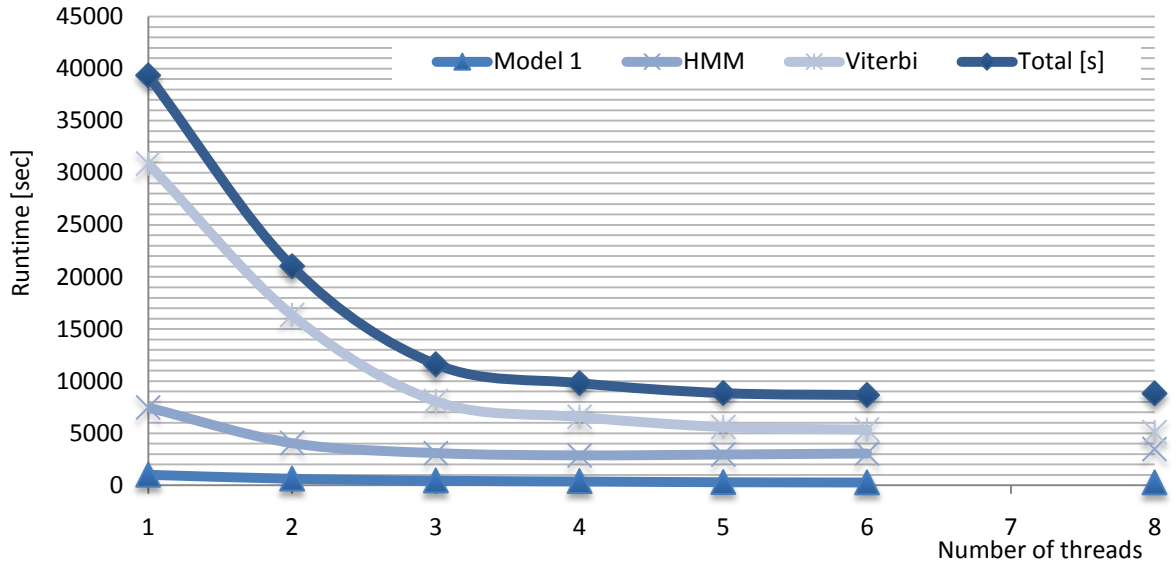


Figure 2.3 MGIZA++ run for CzEng 0.7 on Athena3[19]

Model/threads	1	2	3	4	5	6	7	8
Model 1	1011	635	449	379	305	281		260
HMM	7452	4039	3079	2860	2939	3040		3460
Viterbi	30880	16344	8076	6557	5590	5337		5084
Total [s]	39343	21018	11604	9796	8834	8658		8804
Total [min]	656	350	193	163	147	144		147
Speed-up	100%	53%	29%	25%	22%	22%		22%

Table 2.2 MGIZA++ run for CzEng 0.7 on Athena3

As you can see, the overall speed slightly increases even for NCPUS>4, even though we spare only about 15 minutes. There is, however, another important think to notice. With Kacenka 2 we didn't get above a speed-up of 1:3, whereas here, we almost get a speed-up of 1:5 for 5 CPUs.

2.2.1.4 MGIZA++ on the Acquis Communautaire corpus

The results for the Acquis Communautaire corpus are similar to the CzEng 0.7 corpus (see Figure 2.4). Some of the NCPUS counts have not been tested because it would probably yield no new information.

[19] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/mgiza_tests/czeng_ncpus[1-8]/

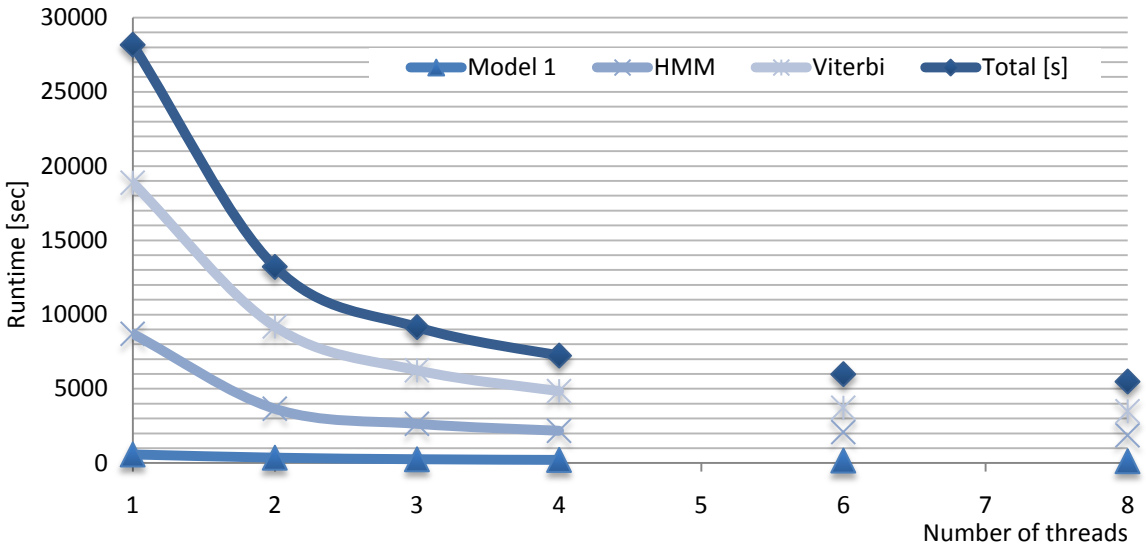


Figure 2.4 MGIZA++ run for the AC corpus on Athena3[20]

Model/threads	1	2	3	4	5	6	7	8
Model 1	585	366	260	215		166		123
HMM	8705	3659	2648	2180		2089		1868
Viterbi	18883	9194	6244	4842		3731		3496
Total [s]	28173	13219	9152	7237		5986		5487
Total [min]	470	220	153	121		100		91
Speed-up	100%	47%	32%	26%		21%		19%

Table 2.3 MGIZA++ run for the AC corpus on Athena3

One thing to notice here is that the speed-up for 8 threads is a little greater than by the CzEng 0.7. We suppose this to be due to the fact that the AC corpus average sentence is longer than that of the CzEng 0.7 corpus.

At this point, we wanted to see the results for AC when run on another server. We used the Athena1 server. The total running time was 432 and 115 minutes for 1 and 4 threads, respectively. The difference is minor, reflecting solely that the Athena1 has slightly more powerful CPUs (each having 2.8 GHz, while Athena3's CPUs each have 2.6 GHz).

2.2.1.5 MGIZA++ on the OpenSubtitles corpus

The OpenSubtitles corpus demonstrates a similar behaviour as Kacenka in that the increasing the number of threads becomes counterproductive at a specific point (here for more than 5 threads).

[20] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/mgiza_tests/acquis_ncpus[1-8]/

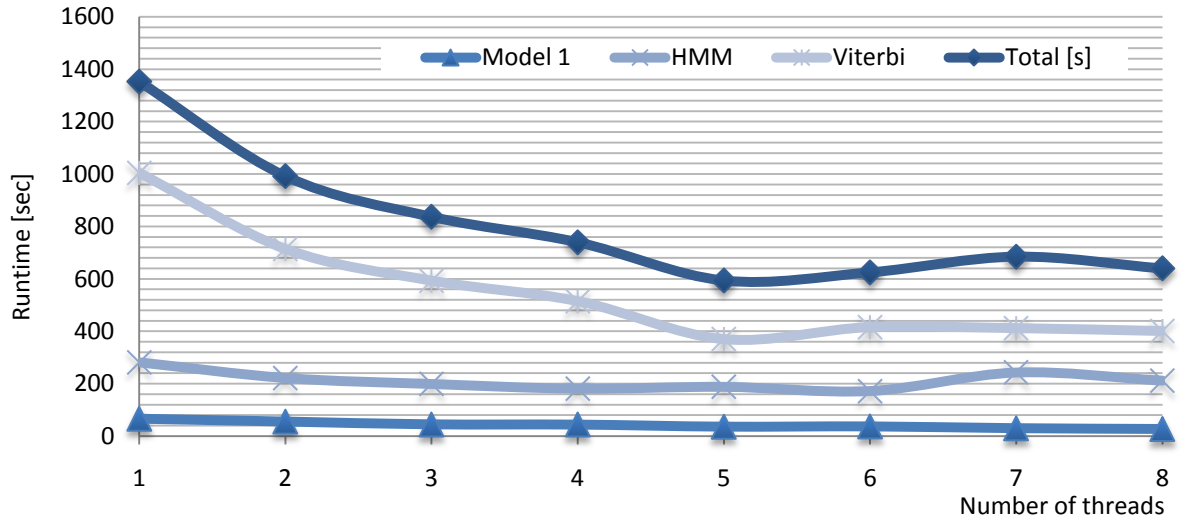


Figure 2.5 MGIZA++ run for the OpenSubtitles corpus on Athena3[21]

Model/threads	1	2	3	4	5	6	7	8
Model 1	67	56	45	44	36	37	30	27
HMM	282	222	199	181	188	172	243	212
Viterbi	1004	714	593	515	370	416	412	401
Total [s]	1353	992	837	740	594	625	685	640
Total [min]	23	17	14	12	10	10	11	11
Speed-up	100%	73%	62%	55%	44%	46%	51%	47%

Table 2.4 MGIZA++ run for the OpenSubtitles corpus on Athena3

To compare the results, we ran the alignment once again on the Athena1 server. The results are depicted in Figure 2.6.

[21] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/mgiza_tests/subtitles_ncpus[1-8]/

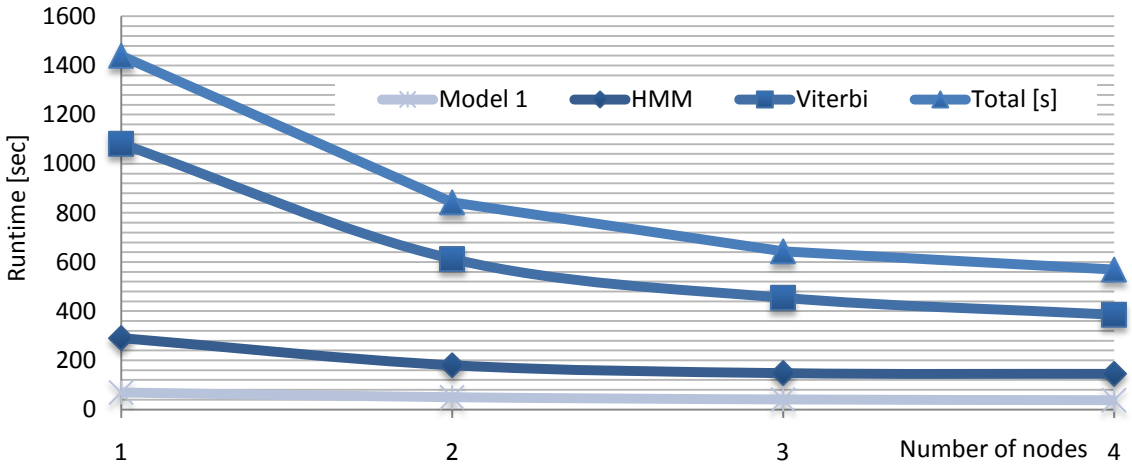


Figure 2.6 MGIZA++ run for the OpenSubtitles corpus on Athena1[22]

Model/threads	1	2	3	4
Model 1	69	50	41	38
HMM	290	180	148	145
Viterbi	1081	613	455	386
Total [s]	1440	843	644	569
Total [min]	24	14	11	9
Speed-up	100%	59%	45%	40%

Table 2.5 MGIZA++ run for the OpenSubtitles corpus on Athena1

The speed-up for 4 CPUs at Athena1 is greater than the speed-up for any number of CPUs at the Athena3 server. Looking just the speed-up in percentage, we could think that Athena1 is considerably quicker compared to Athena3. However, the OpenSubtitles is a small corpus and the whole alignment process takes just minutes so it almost does not matter if we choose Athena1 or Athena3 to align this corpus. The lesson here could rather be that the simpler and shorter the sentences in the corpus are, the less time will the threads spend by computing the individual word alignments and the more often they will access the memory to pop another sentence, potentially blocking other processes wanting more sentences as well.

2.2.1.6 MGIZA++ final notes

During training, the MGIZA++ outputs information on standard output and on standard error output. By analyzing the error output, we found that there are usually sentences whose ratio of its source length and its target length exceeds the allowed ratio (the so called fertility limit, implicit value being 9). For Kacenka 2, this happened 126 times (about 0.1 % of all sentences). For the Acquis Communautaire corpus, this problem did not occur. For the CzEng 0.7 corpus, this problem occurred 19572 times (about 1.7 % of all sentences). For the OpenSubtitles corpus, this happened 11634 times (about 3 % of all sentences).

[22] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/mgiza_tests/subtitles_ncpus[1-4]_athena1/

It seems that the ratio of sentences which exceeded the fertility limit indicates the alignment quality of the source corpora. By looking into the OpenSubtitles corpus at the specific lines where the fertility limit has been exceeded, we found that these sentences are completely misaligned.

2.2.2 PGIZA++

Similar to MGIZA++, PGIZA++ has to be compiled with a GCC of lesser version than 4.3. We compiled it with GCC 4.1.

The PGIZA++ runs on several machines. One machine acts as a master. This machine connects to the other machines via the SSH and coordinates the work of other machines. The master continually checks the work being done by other machines by looking into specific directories where the other machines put their results. These directories have to be shared by all the workstations (using NFS or AFS).

The parallelizing is based on idea that we split the corpus into *n* parts (*n* being the number of nodes in the machine pool ready to run PGIZA++) and do the alignment step in each iteration in parallel. Once all nodes are done with their alignment part, the master takes their work and normalizes the results. This sequence of alignment and normalization is repeated for each iteration.

The advantage of PGIZA++ is its scalability (we can use any number of nodes). However, the I/O can become the bottleneck when the number of child processes is large and also, when the alignment time is much lower than the normalization time (Gao et al., 2008).

2.2.2.1 Training with PGIZA++

The training is run by the train_ega.sh script (available at Gao, 2009). This script first runs the snt2plain, plain2snt, and mkcls tools, after which the training itself is launched.

We tested the PGIZA++ performance on these servers: athena[1|2|3], minerva1, pcnlp[3|4|5|6]. The athena[1|2] server each offers 4x2.8 GHz, the athena3 has 8 CPUs, each having 2.6 GHz. The minerva1 server has 4x2.33 GHz. The pcnlp[3|4|5|6] server each has 2x2.66 GHz.

Table 2.6 gives an overview of the nodes used for the testing.

Nodes	Master	Other nodes						
2	Athena2	Pcnlp4						
4	Athena2	Pcnlp4	Pcnlp5	Pcnlp6				
6	Athena2	Athena1	Pcnlp3	Pcnlp4	Pcnlp5	Pcnlp6		
8	Athena2	Athena1	Athena3	Minerva1	Pcnlp3	Pcnlp4	Pcnlp5	Pcnlp6

Table 2.6 Computers used for testing PGIZA++

2.2.2.2 PGIZA++ on the Kacenka 2 corpus

Kacenka 2 is a very small corpus and after reading the preliminary notes on PGIZA++ it should be clear that this corpus is not suitable for PGIZA++. To prove this, we ran the alignment (see Figure 2.7).

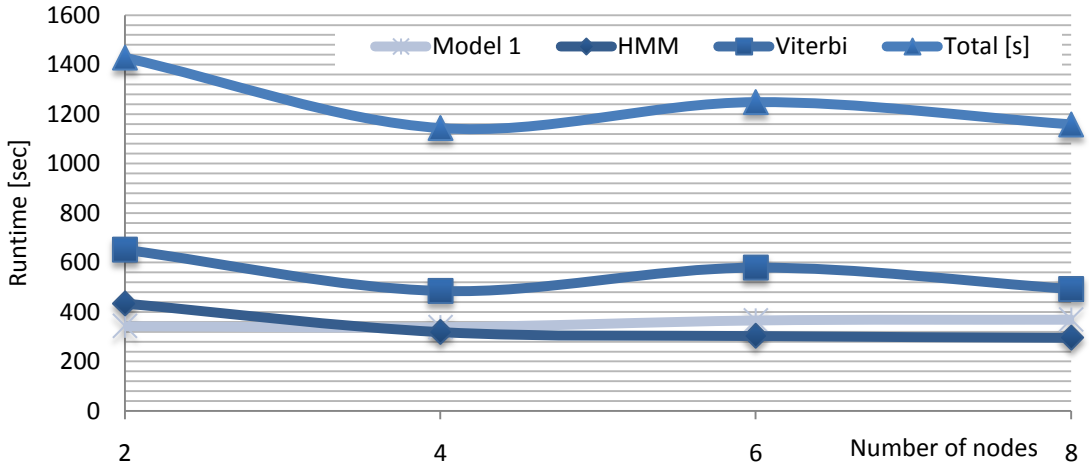


Figure 2.7 PGIZA++ run for the Kacenka 2 corpus[23]

Model/nodes	2	4	6	8
Model 1	343	339	366	369
HMM	434	319	303	296
Viterbi	651	485	579	493
Total [s]	1428	1143	1248	1158
Total [min]	24	19	21	19
Speed-up	88%	70%	77%	71%

Table 2.7 PGIZA++ run for the Kacenka 2 corpus

You can check for yourself that the speed is worse compared to MGIZA++. What is more, using more than 4 nodes takes more time than using just 2 or 4 nodes.

2.2.2.3 PGIZA++ on the CzEng 0.7 corpus

We attempted to train the CzEng 0.7 corpus on PGIZA++, first with 4 nodes and then with 6 nodes but each time the training failed because one machine failed. When we switched the machines then another machine failed so the problem is not the server selected. Looking into the log files we found the following error:

```
In source portion of the training corpus, only 1 unique tokens appeared
In target portion of the training corpus, only 165370 unique tokens appeared
```

[23] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/pgiza_tests/kacenka_nodes[2|4|6|8]/

It seems that there is a problem with reading the data. To further investigate on the cause of this error, we repeated the test for the Books 2 corpus. We ran the test on 4 nodes but again, the same error caused a premature termination of the training process. The server and the iteration number differed from the server and iteration number where the error occurred for CzEng 0.7.

To find the cause for this error, we would have to examine thoroughly how the bash scripts used for the training process work. Unfortunately, due to time constraints, we had to leave this problem unresolved.

2.2.2.4 PGIZA++ on the Acquis Communautaire corpus

Does the PGIZA++ bring a speed-up when given the AC on input? We ran PGIZA++ for 2, 4, 6, and 8 nodes and found that it yields better results than the MGIZA++ indeed (see Figure 2.8).

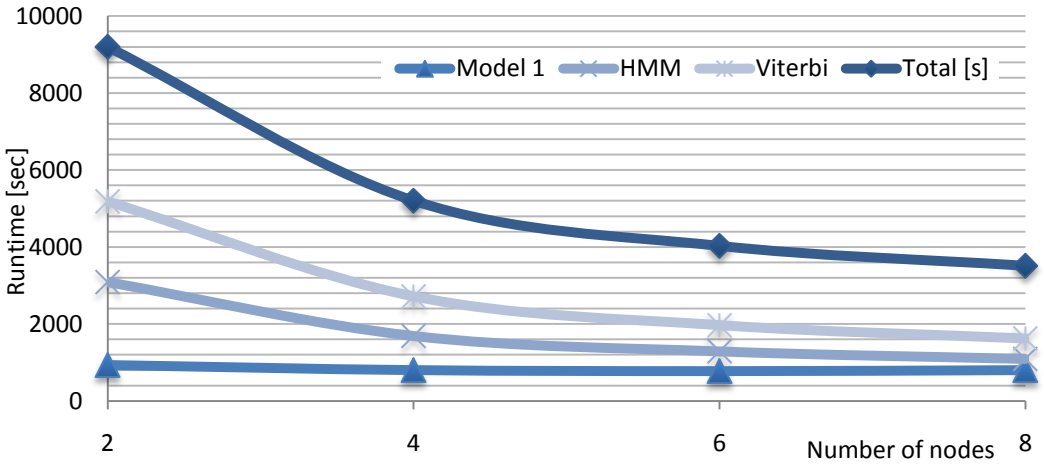


Figure 2.8 PGIZA++ run for the AC corpus[24]

Model/threads	2	4	6	8
Model 1	929	798	772	797
HMM	3088	1689	1288	1092
Viterbi	5183	2716	1967	1620
Total [s]	9200	5203	4027	3509
Total [min]	153	87	67	58
Speed-up	33%	18%	14%	12%

Table 2.8 PGIZA++ run for the AC corpus

This is the first time (and, alas, the last time too) we see the PGIZA++ outperform the MGIZA++. We can tentatively imply that if we have a corpus of the size of AC or greater we should start considering to choose PGIZA++ over MGIZA++. Of course, we would have to support this

[24] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/pgiza_tests/acquis_nodes[2|4|6|8]/

conjecture by other tests because it may be the high average sentence length rather than the overall size of the AC corpus that gives the PGIZA++ advantage over the MGIZA++.

2.2.2.5 PGIZA++ on the OpenSubtitles corpus

To make the testing complete, we also ran PGIZA++ on the OpenSubtitles corpus (see Figure 2.9). The results are not dissimilar to the Kacenka's results. What is worth noting is that the speed-up in percentages is exactly the same compared to PGIZA++ run on Kacenka. The Viterbi alignment, for some reason, takes consistently more time on 6 nodes rather than on 4 or 8 nodes for small corpora.

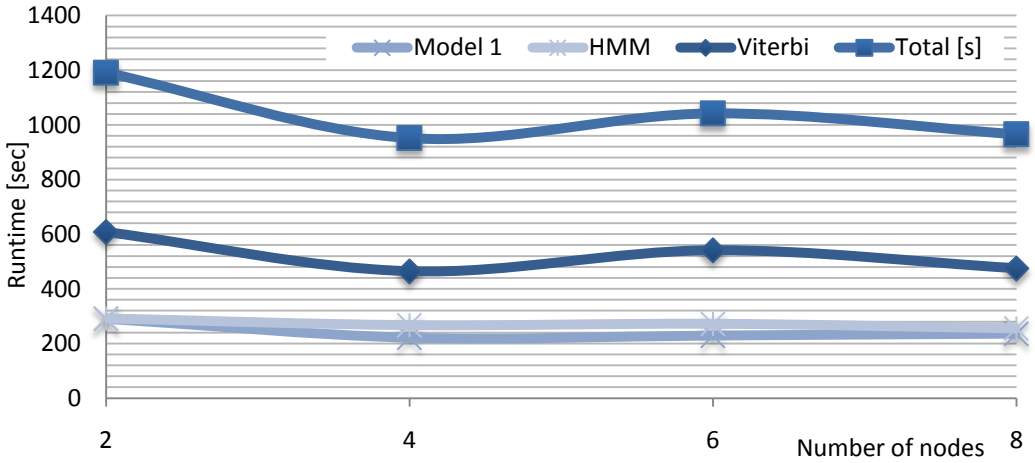


Figure 2.9 PGIZA++ run for the OpenSubtitles corpus[25]

Model/nodes	2	4	6	8
Model 1	291	221	229	236
HMM	292	267	272	255
Viterbi	608	464	541	475
Total [s]	1191	952	1042	966
Total [min]	20	16	17	16
Speed-up	88%	70%	77%	71%

Table 2.9 PGIZA++ run for the OpenSubtitles corpus

2.2.3 Should we use MGIZA++ or PGIZA++?

To sum the above sections up, it is highly advisable to use MGIZA++ on a small corpus like Kacenka or OpenSubtitles (Figure 2.10 and Figure 2.11). On the other hand, when training on a larger corpus like Acquis Communautaire, the PGIZA++ seems a better choice (Figure 2.12).

When using MGIZA++, we found that in most cases it does not really matter if we choose Athena3 with its 8 CPUs or Athena1 with its 4 CPUs. The training time difference between Athena3 (8 CPUs) and Athena1 (4 CPUs) is maximally tens of minutes for the corpora tested.

[25] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/pgiza_tests/subtitles_nodes[2|4|6|8]/

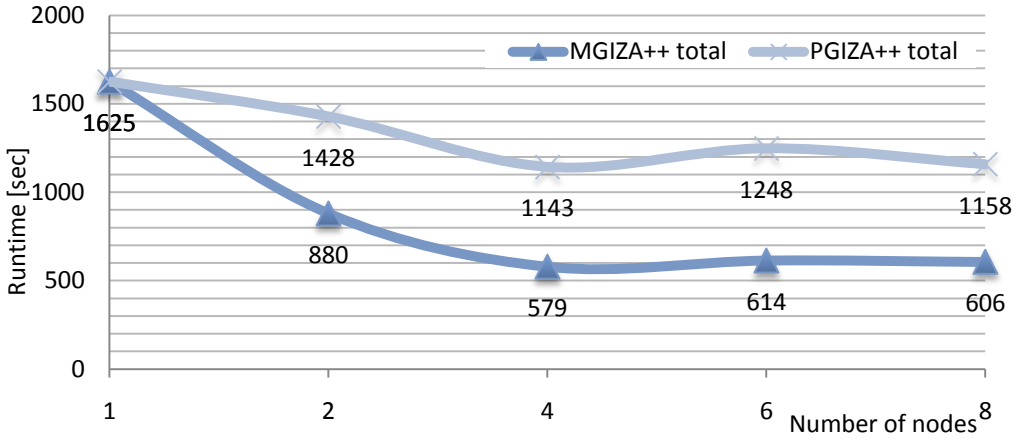


Figure 2.10 MGIZA++/PGIZA++ comparison for the Kacenka corpus

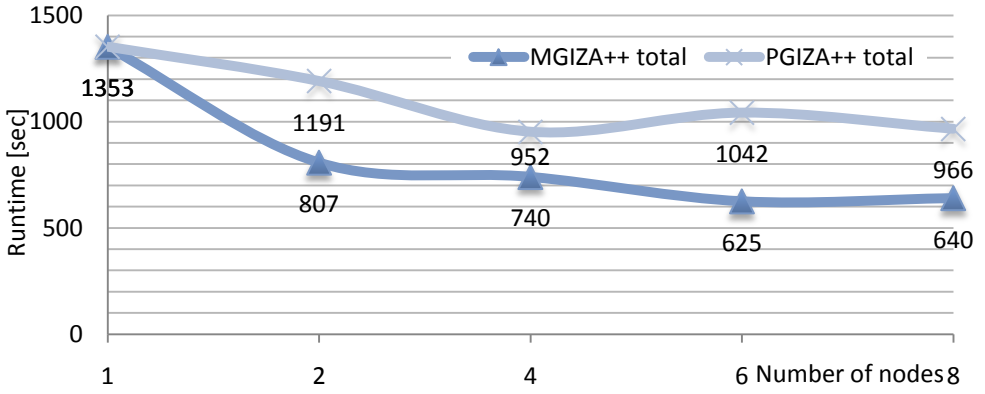


Figure 2.11 MGIZA++/PGIZA++ comparison for the OpenSubtitles corpus

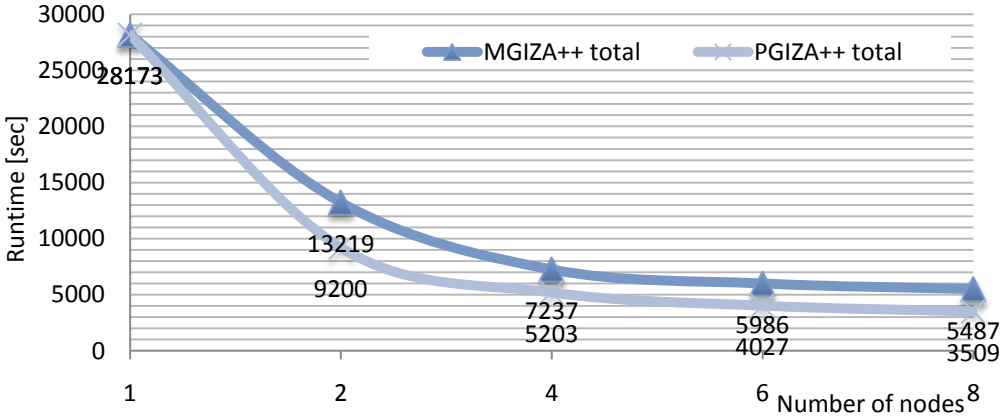


Figure 2.12 MGIZA++/PGIZA++ comparison for the AC corpus

2.3 Assembling a machine translation system

In chapter 1 we outlined the core parts that a statistical machine translation system consists of – parallel corpora, language model, translation model, decoder and evaluator. So far, we have thoroughly analyzed how to create word alignment using the GIZA++ tool. We have also seen that GIZA++ requires other programmes to pre-process the data, such as mkcls, developed by Franz Josef Och. If we investigate further we learn that Och implemented extensions for GIZA++, enhancing the capabilities of the older GIZA, which in turn was developed in 1999 by a team at Johns-Hopkins University (Och, 2001). Later, Qin Gao took the GIZA++ and added support for multithreading.

The idea we are trying to convey is that different parts of a machine translation system are being developed by different groups of people and that it is quite common to enhance the capabilities of existing programmes, rather than implement them anew from scratch.

A similar approach is usually taken when building the entire machine translation system - existing blocks are utilized and integrated. Popular machine translation systems, such as Moses (Euromatrix Project, 2008a), Joshua (Callison-Burch, 2009) or Cunei (Phillips et al., 2009) follow such a modular architecture.

In this thesis, we decided to base our following work on the Moses translation system. We did this because the system is open source, well documented and there is a lot of active development going on. Moreover, Philipp Koehn, under whose guidance the project is being developed, claims that the system is "the de facto benchmark for research in the field" (2010).

2.3.1 Architecture of the Moses translation system

The first thing to notice is that the term *Moses* is used both for the entire statistical machine translation system as well as for the decoder, which is only a part of the system. In the following text, we will either say *Moses system* or *Moses decoder* to make the distinction clear.

To start with, inspect Figure 2.13 to see the top-level architecture of the Moses system. Note that boxes with red border indicate that the part has not been developed yet (as of May 2010).

As you can see, the **input** to the system can either be a plain text, XML, a confusion network or a lattice. We will be using the first option because our corpora are in plain text[26]. Another thing to mention here is that we will be doing phrase-based translation that is not factored. That means that we will provide no analytical or morphological information (factor) in the input corpus except for the literal words.

Next in the diagram we can see the **translation model**. As we already mentioned in section 1.2.2, the main part of the translation model is the phrase-translation table. Moses creates this table from the output of GIZA++. It is quite common that the phrase-translation table is very large (up to several tens of gigabytes) to fit into memory, in which case we may need to load the

[26] The remaining input options are used for hierarchical syntax-based tree models and/or for models that integrate machine translation with other upstream speech processing tools, such as speech recognizers.

table from disk. We will be loading the phrase-translation table both from memory and from hard disk, based on its size.

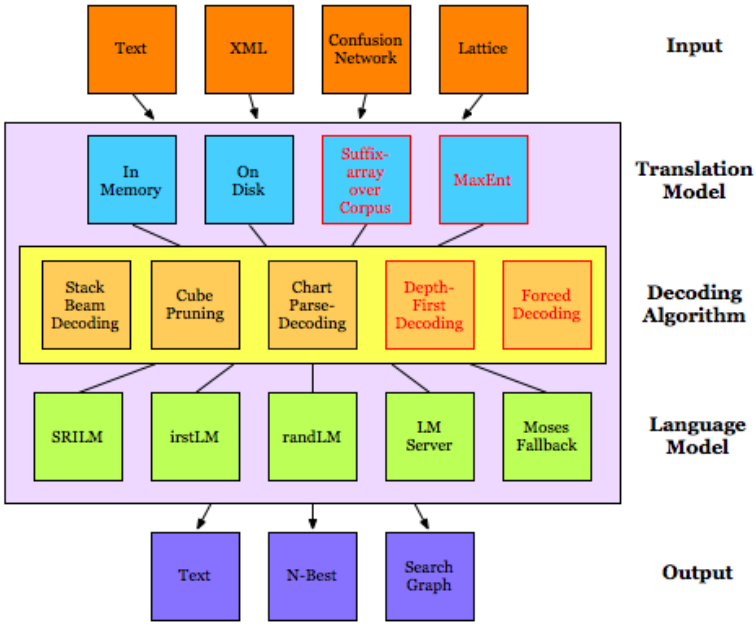


Figure 2.13 Modular architecture of the Moses system (Koehn, 2010)

For the **decoding**, we will be using the main version of the Moses decoder which is based on stack beam decoding[27].

The **language model** can be based on several third party toolkits – SRILM, irstLM and randLM. In this thesis we work exclusively with the SRILM toolkit.

The **output** of the system is usually the translated plain text. However, if we want to peak into the internal workings of the decoder, we may choose the N-Best or Search Graph options. In the first case we learn what other hypotheses (output sentences) were evaluated (but eventually received lower score that the winning hypothesis). In the latter case, we get a dump of the search graph.

2.3.2 Methodology of training and testing with the Moses system

In this section we dig deeper into the Moses system. We introduce the moses.py[28] script that we created for training and evaluating various versions of the system.

The moses.py script has been created with the objective to compare performance of various modifications of the Moses system. We will run the system many times, each time modifying one

[27] Cube pruning is another search algorithm that is "faster than the traditional search at comparable levels of search error" (Euromatrix Project, 2010b). Chart Parse-Decoding is used for tree-based decoding.
[28] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/

input variable (independent variable), and then evaluate the performance measured in BLEU (dependent variable). We want to answer these questions:

a) How does the size of the training corpus influence the quality of the translation?
b) Will the quality of the translation increase if we incorporate a bilingual dictionary into the training data?
c) How will the incorporation of an extra monolingual corpus into the language model affect translation quality?
d) What effects will have a morphological pre-processing of the Czech part of the corpus?

2.3.2.1 Preparing the corpora

The corpora we will use[29] have first to be split up into three parts – the **training**, **development** and **test sets**. The division follows the ratio 90:5:5. We split the corpus based on the count of the sentences. In case we combine two corpora to train either the translation or language model, the training data is a concatenation of the training sets of both corpora, while the development and test sets are taken only from one of the corpora (the one we are interested to analyze primarily).

Before we start training, we first **tokenize**, **filter**, **clean**, **pre-process** and **lowercase** the input corpus[30]. By filtering we mean discarding sentences that are longer than 40 words[31]. By cleaning we mean running a script[32] that erases quotation marks and apostrophes that indicate direct speech (clitics remain unaffected). By pre-processing we mean converting the words in the input corpus into their lemmas or other ways of adding or replacing words in the corpus with the intent to convey some extra morphological information (see section 2.3.2.2 for more details). Please note that we still work with unfactored systems. Finally, lowercasing the input corpus is required.

2.3.2.2 Morphological pre-processing

As already mentioned in section 1.2.5.1, we will be using both the Prague Dependency Treebank (PDT) as well as the Libma library to pre-process the Czech part of the corpus. We base our work on the findings of Goldwater et al. (2005). We will test three scenarios:

a) We will replace all words with their lemmas.
b) We will do the same as in point a) but we also add some extra pseudo words into the corpus.
c) We will do the same as in point b) but only for words that appear sparsely in the input corpus. Words that occur often in the corpus will be completely unaffected.

In the **first scenario**, we lemmatize all words. This will clearly discard some information but at the same time, it should ameliorate the effects of data sparseness. To give an example, the sentence "*Jen ať tam* jde *děda." ("Let the old man* go*.")* will be converted to *"Jen ať tam* jít *děda."*

[29] Stored jointly in the minerva1:/mnt/minerva1/nlp/projects/mt/corpora/ directory.
[30] This step corresponds to step 1 in the moses.py script.
[31] More specifically, only training data used for building the translation model are filtered. This is because GIZA++ run takes a long time on unfiltered data. Also note that development and test data are not filtered.
[32] Located at minerva1:/mnt/minerva1/nlp/projects/mt/tools/myown/corpora preparation/myclean.py

In the **second scenario**, apart from lemmatization we do two more modifications: In case it is a noun, we indicate whether it is singular or plural number (e.g., *house* will correspond to *dům+S*, while *houses* will correspond to *dům+P*). In case the word is a verb, we indicate the person and tense. For instance, the sentence "*Jen ať tam jde děda."* will be converted to *"Jen ať tam PER_3 jít+TEN_P děda."* We indicate that the verb *jít (go)* is in the present tense *(TEN_P)* and in the third person *(PER_3)*. Also note that the person indication is a separate word so as to imitate the pronoun *(he, she, it)* that is often omitted in the Czech language. In contrast, the tense indication *(TEN_P)* is concatenated with the base form *jít (go)* so as to imitate the (quite simple) inflectional morphology of English verbs *(go* vs. *goes)*.

In the **third scenario**, we apply the modifications described in the second scenario but this time only for words that occur in the corpus with a frequency lower than a defined threshold. We will work with the threshold of 50. It follows that before the actual pre-processing, the corpus must be analyzed, frequent words extracted and stored somewhere[33]. Later, we will have to cross-check each word, determine if it is a frequent word or a sparse word and either carry out scenario 2 or leave the original word form unaltered.

Automatic morphological annotation is not as unproblematic as it may seem. We already touched upon the problem of ambiguity of word forms and different quality of automatic annotators in section 1.2.5.1. The **Libma library**, for example, does not analyze the neighbourhood of the word because you can give it only one word form on input, not the whole sentence. If there are more lemmas or more options of part-of-speech tags for a given lemma, it will return all of them. The **PDT**, on the other hand, takes whole sentences on input and it does analyze the neighbourhood of the word. In case there are more lemmas corresponding to a word form, it will probably return the correct one as the first lemma and also inform you about the alternative lemmas.

How should we go about when more than one lemma is found? After some preliminary testing (on the same corpora which we will be using for main testing) we discovered that it is better to do the lemmatization only in case an unambiguous lemma is found by the morphological analyzer. If there are several matches of the same lemma but multiple parts of speech returned[34], we will do the morphological pre-processing with the first part of speech returned.

2.3.2.3 Building the language and translation model[35]

The **language model** will be created with the SRILM toolkit[36] from a tokenized and lowercased training set of the given corpus or combination of more corpora. We will create n-grams up to order 5. We will use interpolation and Kneser-Ney discounting.

The training of the **translation model** takes place in several steps[37]. First of all, we pre-process the corpus with the plain2snt and mkcls tools[38]. Then we run MGIZA++ to get word alignments[39].

[33] This is done by the frequent_words.py script.
[34] Refer to 1.2.5.1 where we did not know for sure if the word form is in accusative or vocative.
[35] These steps correspond to steps 2 and 3 in the created moses.py script.
[36] More specifically, we will use the ngram-count executable.

Based on our analysis of MGIZA++, we decided to run MGIZA++ in 4 threads. Moreover, we need to run MGIZA++ twice (both in the Czech-English and English-Czech direction; refer to 1.2.2.1). In order to spare time, we will run both directions in parallel. In this way, there will be up to 8 threads running at a moment, computing word alignments.

Once the MGIZA++ runs finish, we compute a final word alignment taking into account the two alignments from both runs of MGIZA++. There are several options how to combine the two alignments. We will use the default heuristic called *grow-diag-final*[40]. It starts with the intersection of the two alignments and then adds additional alignment points (Koehn, 2010). Figure 2.14 shows an example of the alignment.

```
0-0 1-1 2-2 3-3 3-4 4-4 5-5 6-6 6-7 7-8
tři prsteny pro krále elfů pod nebem ,
three rings for the elven-kings under the sky ,
```

Figure 2.14 Example of word alignment

Having the word alignment of the entire corpus, the Moses system uses it to extract a lexical translation table[41]. An extract from an example translation table is shown in Figure 2.15.

```
king král 0.5250000
the král 0.1750000
of král 0.0500000
nargothrond král 0.0250000
elven-king král 0.0250000
elf-kings král 0.0250000
```

Figure 2.15 Example of a translation table

Next, all phrases are extracted and dumped into a file[42] (see extract of this file in Figure 2.16).

```
krále elfů ||| the elven-kings ||| 0-0 0-1 1-1
krále elfů pod ||| the elven-kings under ||| 0-0 0-1 1-1 2-2
pro krále elfů ||| for the elven-kings ||| 0-0 1-1 1-2 2-2
pro krále elfů pod ||| for the elven-kings under ||| 0-0 1-1 1-2 2-2 3-3
```

Figure 2.16 Example of extracted phrases

---

[37] The entire training is done by the train-factored-phrase-model.perl script, which is part of the Moses translation system.
[38] Already explained in section 2.2.1.1.
[39] Please note that we will not be using PGIZA++. Although PGIZA++ proved to perform better on larger corpora than MGIZA++ (see section 2.2.2.4), the Moses system provides considerably easier integration with MGIZA++. Moreover, MGIZA++ has lower synchronization overhead than PGIZA++.
[40] The output of this step can be found in the aligned.grow-diag-final-and files located at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_*/model/ directories.
[41] These files are named lex.e2f and lex.f2e and are in the model directories as well.
[42] More specifically, there are three files under the implicit settings - extract.gz, extract.inv.gz, and extract.o.gz. The first two are the base and inverse version of what is shown in Figure 2.16, while the third is created when a lexicalized reordering model is trained.

In the next step, all phrases are scored (see Figure 2.17). There are five scores in the file: phrase translation probability $\varphi(f|e)$, lexical weighting $lex(f|e)$, phrase translation probability$\varphi(e|f)$, lexical weighting $lex(e|f)$ and phrase penalty.

```
králové ||| kings ||| 0.416667  0.228571  1         0.727273 2.718
králů   ||| kings ||| 0.166667  0.228571  0.666667 0.470588 2.718
králi   ||| kings ||| 0.0833333 0.0571429 0.333333 0.153846 2.718
```

Figure 2.17 Example of a scored phrase table

The last step is to build the reordering model. We will use the *msd-bidirectional-fe* option to build the reordering model. This reordering is an addition to the standard reordering model which gives cost linear to the reordering distance (recall the distortion probability from section 1.2.2).

Finally, Moses[43] stores the information about the models created so far into the moses.ini file. It contains information on where the models are stored and their parameters. Later, this file is used by the decoder when doing the actual translation.

2.3.2.4 Tuning and testing[44]

If we look into the abovementioned moses.ini file we will see that it contains default weights that the decoder uses when evaluating the most probable translation of a given sentence. Figure 2.18 displays an extract from a simple moses.ini file.

```
# distortion (reordering) weight
[weight-d]
1

# language model weights
[weight-l]
1

# translation model weights
[weight-t]
1

# word penalty
[weight-w]
-1
```

Figure 2.18 Extract from a simple moses.ini file[45]

For each sentence, the decoder has to evaluate the probability:

[43] Specifically, the train-factored-phrase-model.perl script.
[44] Tuning and testing corresponds to steps 4,5,6,7 in the moses.py script.
[45] The distortion and translation weights are actually vectors (implicitly of orders 7 and 5, respectively) but our intent is here to keep the example simple. However, if you inspected the weights thoroughly, you would learn that 5 translation weights correspond to the five weights for each phrase listed in figure 2.15.

$$p(e|f) = \phi(f|e)^{weight_\phi} \times LM^{weight_{LM}} \times D(e,f)^{weight_d} \times W(e)^{weight_W} \quad (2.1)$$

Now if you recall equations 1.2 and 1.3, you will see that there is almost nothing new in equation 2.1. We just added a fourth member $W(e)$ which is the word penalty that ensures that the translations do not get too long or too short. Then we raised each member of the product to a weight which we found in the moses.ini configuration file.

It should now be clear what the purpose of tuning is. We give the decoder a set of previously unseen sentence pairs (the development set) and it iteratively adjusts the weights in the moses.ini file so that the resulting BLEU score of the development set is maximal. Whether it will eventually improve the BLEU score of the testing set, that remains a question.

Next, we run the **decoder** on the test set. On input, the decoder requires the moses.ini configuration file and the text to be translated. The decoder then loads the language model, phrase table and reordering table into memory and starts translating. Often, however, the models are too big to fit into memory so it is usual to filter the models first[46]. Filtering means that the model will be reduced to contain only phrases that occur in the test set.[47]

2.3.2.5 Evaluation[48]

The evaluation was performed with the multi-bleu.perl script on the output of the decoder and the reference translation. Both these texts are still lowercased at the moment of evaluation.

[46] Using the filter-model-given-input.pl script, which is part of the Moses system package.

[47] During our following experiments, the filtering alone was not sufficient – the decoder still needed more than 4 GB memory and once this memory threshold was exceeded, it received SIGABRT and the translating could not start. This happened despite the fact that the Athena 3 server hosts 64 GB memory. After an investigation we attempted to compile the 64 bit version of the Moses decoder but the compilation failed. Finally, we found out that the filtered model can be binarized with the script filter-and-binarize-model-given-input.pl. After this step, the decoder worked within the limits of the 4 GB memory for all our experiments.

[48] Evaluation is performed in step 9 of the moses.py script.

3 Analysis of the created SMT system

In this final chapter we will first analyze how various corpora characteristics and the degree of morphological pre-processing influence the resulting BLEU. Each time, we start with a base system and change one variable to see the effects on BLEU. Once we gain some insight, we proceed to train a final system which is inspired by the Translation Task from WMT 10 (European Commission, 2010a).

3.1 Analysis of individual factors

3.1.1 Size of the corpus

In this section, we will analyze the relation between corpus size and the resulting BLEU score. First, we will work with the Kacenka corpus. We start with evaluating the base BLEU for the Kacenka corpus, which is the BLEU we get when we train both the language and translation model from the training data of the corpus. We do no morphological pre-processing and no tuning[49].

Next we successively truncate the training set to the first 10 %, 20 %, 30 %, …, 90 % sentences and repeat the steps executed for the base BLEU (both language and translation model will be trained on only a part of the training data). The testing set will be identical for all 10 measurements (it will not be truncated). The development set will not be used at all.

Figure 3.1 shows the results. We can see that the relation is roughly linear, with a slight skew from the trend in the area around 50 %.

Now we repeat this experiment for the Acquis corpus. Recall that Acquis has about twice as many sentences as Kacenka and the average sentence is also twice as long compared to Kacenka. However, many of the sentences in Acquis will be filtered out in the first step. Table 3.1 and Table 3.2 display the different characteristics of both corpora.

[49] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_kacenka_base/

% used	BLEU	Sentences	Unique CS tokens	Unique EN tokens
10%	9,20	10290	24632	10660
20%	10,73	20644	38330	15024
30%	11,99	30919	48641	18161
40%	12,88	41141	57004	20622
50%	12,93	51422	65759	23990
60%	13,74	61559	74613	27295
70%	14,81	71798	82442	30248
80%	15,52	82136	88426	32411
90%	16,54	92550	93728	34267
100%	17,71	102944	97714	35603

Table 3.1 Relation between corpus size and BLEU (Kacenka)[50]

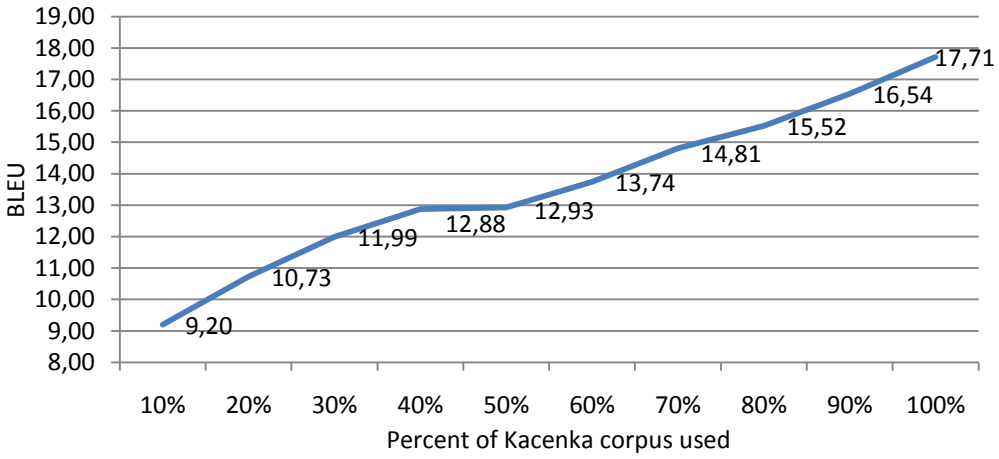


Figure 3.1 Relation between the size of the Kacenka corpus and the resulting BLEU[51]

% used	BLEU	Sentences	Unique CS tokens	Unique EN tokens
10%	34,62	14816	13011	7402
20%	37,44	30619	20832	11381
30%	39,39	45975	26974	14306
40%	41,73	61233	31195	16168
50%	43,90	76192	35307	18318
60%	45,00	90815	39223	20306
70%	46,17	105985	42178	21694
80%	47,14	120769	45325	23290
90%	48,09	136082	48101	24844
100%	48,89	150770	50312	25893

Table 3.2 Relation between corpus size and BLEU (Acquis)[52]

[50] These numbers come from the training sets after filtering, cleaning and tokenization.
[51] The relevant work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_kacenka_part_*xx* where *xx* is 10, 20, 30, 40, 50, 60, 70, 80, and 90.

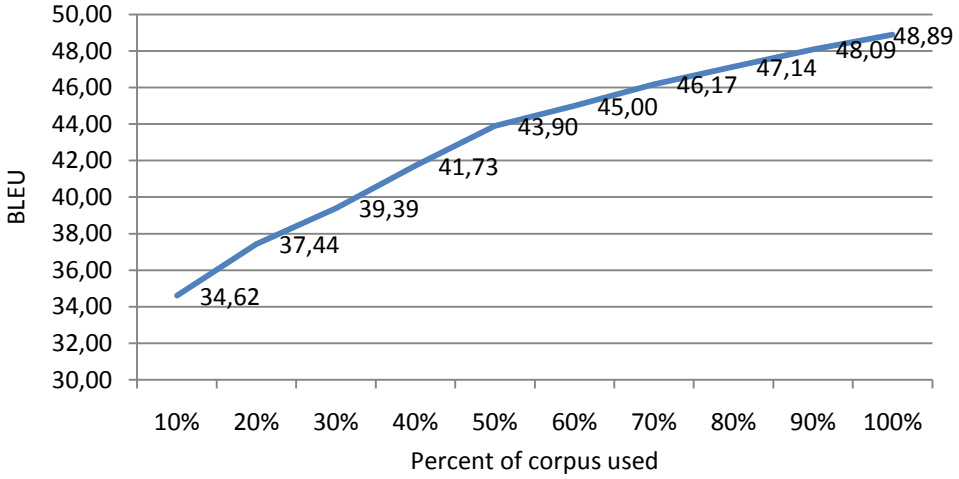


Figure 3.2 Relation between the size of the Acquis corpus and the resulting BLEU[53]

Figure 3.2 shows the results for the Acquis corpus. We can see that the relation is linear as well but the slope is sharper, especially in the first 50 %. The resulting BLEU is considerably higher than the BLEU for Kacenka. This agrees with a simple intuition that fiction uses much richer language and the translation is more difficult. The numbers of token types confirm this. The Kacenka corpus has almost 100000 unique Czech tokens, while Acquis has only about 50000.

3.1.2 Additional training data

In this section, we first investigate if the incorporation of a bilingual Czech-English dictionary into the training data improves the BLEU. We do this both for the Lite Dict corpus and Full Dict corpora. The training data is simply a concatenation of both the Kacenka's training set and the respective dictionary. Figure 3.3 shows the results.

As you can see, including a simple dictionary resulted in BLEU increase of 0.3 %. It may be that the testing set contains previously unseen words (the language of fiction is rich).

Again, we will run this test for the Acquis corpus as well (see Figure 3.4). Surprisingly, the effects are quite different – Acquis benefitted from Full Dict containing additional technical phrases, while the effects of Lite Dict are close to none.

[52] These numbers come from the training sets after filtering, cleaning and tokenization.
[53] The relevant work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_acquis_part_*xx* where *xx* is 10, 20, 30, 40, 50, 60, 70, 80, and 90.

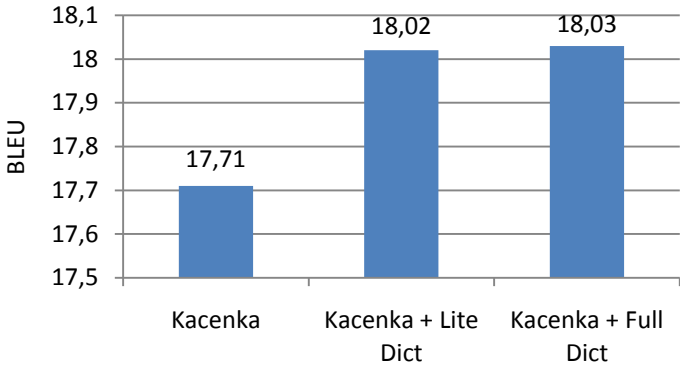


Figure 3.3 Adding a dictionary into the training data of the Kacenka corpus[54]

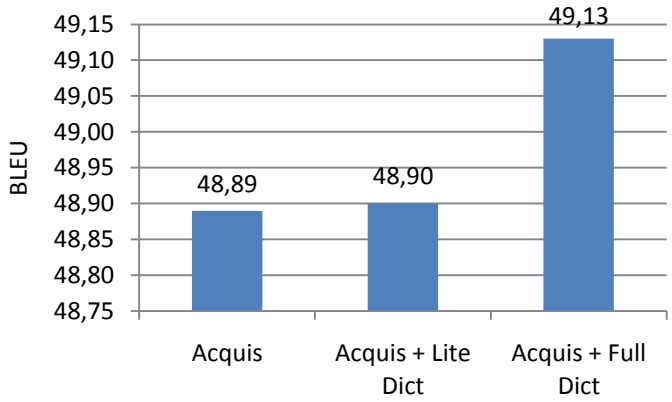


Figure 3.4 Adding a dictionary into the training data of the Acquis corpus[55]

Now we focus on the training data for the language model. We start with the base scenario in which the language model is trained on the Kacenka training set. Next, we train the language model on different corpora – first, the Books 2 corpus and then the CzEng 0.7 corpus. In these two scenarios, the Kacenka corpus is not used to build the language model at all. Next, we repeat the same two scenarios but this time, we add the Kacenka corpus so that we will train on Kacenka + Books 2 and then on Kacenka + CzEng 0.7. The results are shown in Figure 3.5.

[54] The working directories are to be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_kacenka_dict_[lite|full]/
[55] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_acquis_dict_[lite|full]/

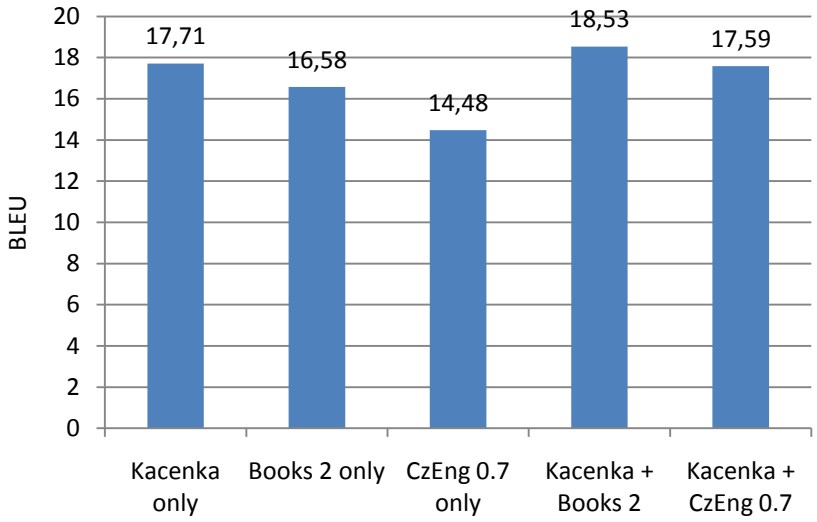


Figure 3.5 Training language model on various corpora combinations[56]

We can clearly see that eliminating the Kacenka corpus from the language model training data is counterproductive. However, we can see that working with smaller Books 2 corpus still yields better results than the CzEng 0.7 corpus. The genre of the corpus is more important than its size.

The last two columns convey a similar message: it is better to combine Kacenka with a fiction corpus, not the multi-domain CzEng 0.7 (the BLEU in this last case still drops slightly).

3.1.3 Morphological pre-processing

As already indicated in section 2.3.2.2, we will test three scenarios of morphological pre-processing, each time using both the Libma library[57] and the Prague Dependency Treebank[58].

Both Libma and PDT are exploited through a Python interface[59]. The interface for Libma is more sophisticated and allows us to set up several parameters. We will set the case sensitivity to 0 and lemmatization level[60] to 111. These quite restrictive settings should have the effect that we do

[56] The work for the five scenarios can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/ in the following directories: work_kacenka_base, work_kacenka_lm_books2, work_kacenka_lm_czeng1, work_kacenka_lm_books2_kacenka, and work_kacenka_lm_czeng1_kacenka

[57] The pre-processing is done by two our scripts – morphology_ma.py and morphology_ma_env.py

[58] The pre-processing is done by the morphology_pdt.py script.

[59] Libma through minerva1:/mnt/minerva1/nlp/projects/mt/tools/ma/libma/pylibma/. PDT through minerva1:/mnt/minerva1/nlp/local/lib/python2.5/site-packages/.

[60] The SetLemmatization() method.

not unnecessarily discard much information[61]. The PDT Python interface does not allow us to set any level of lemmatization[62].

Let's see the results (Figure 3.6).

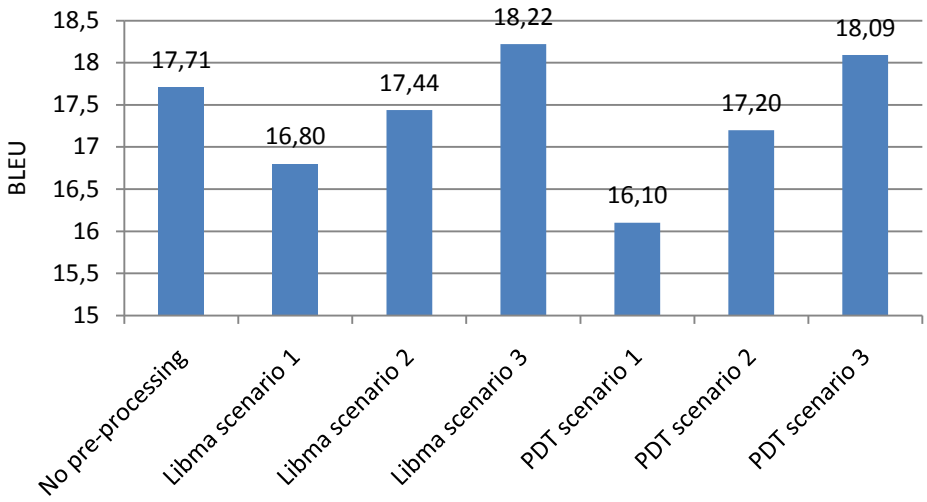


Figure 3.6 Morphological pre-processing of the Kacenka corpus[63]

The results of the first two scenarios are rather disappointing – the BLEU drops for both Libma and PDT. However, the third scenario does improve the BLEU, in the best case from 17.71 to 18.22. We can also see that the Libma library performs better in all three scenarios. But why? Let us see the extent to which the morphological pre-processing reduces the number of unique tokens in the Czech portion of the corpus (Figure 3.7).

[61] For example, *nedobrý* (not good) will not be stripped to *dobrý* (good). *Nejkrásnější* (most beautiful) will not be stripped to *krásný* (beautiful). Generally, negation, superlatives, and other prefixes won't get lost.
[62] In retrospect, we discovered that the PDT's positional tags 10 (grade) and 11 (negation) could possibly be used in an analogous way to the Libma's SetLemmatization() method. However, further tests would have to be run to confirm whether PDT's performance would exceed Libma's performance if these tags were exploited.
[63] The relevant working directories can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_kacenka_[libma|pdt][1|2|3]/

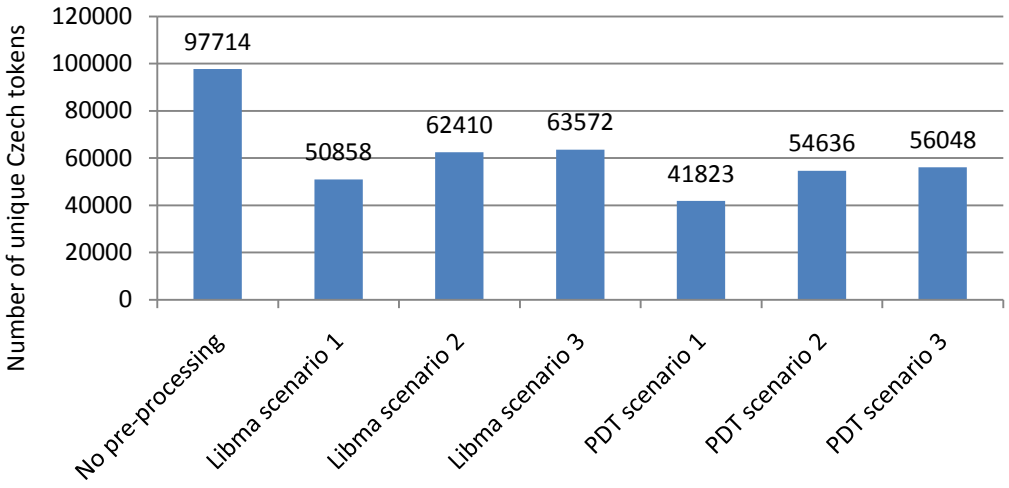


Figure 3.7 Number of unique Czech tokens in the training set of Kacenka after morphological pre-processing

We can see that the Libma library is more conservative when doing lemmatization. In the pre-processed corpus, there are several thousand lemmas more for each scenario compared to the PDT pre-processing. It may be that the PDT drops too much information when doing the lemmatization.

We will run the morphological pre-processing for the Acquis corpus as well to see if there is a difference. Figure 3.8 shows the resulting BLEU under individual scenarios. Figure 3.9 displays the degree of unique tokens reduction.

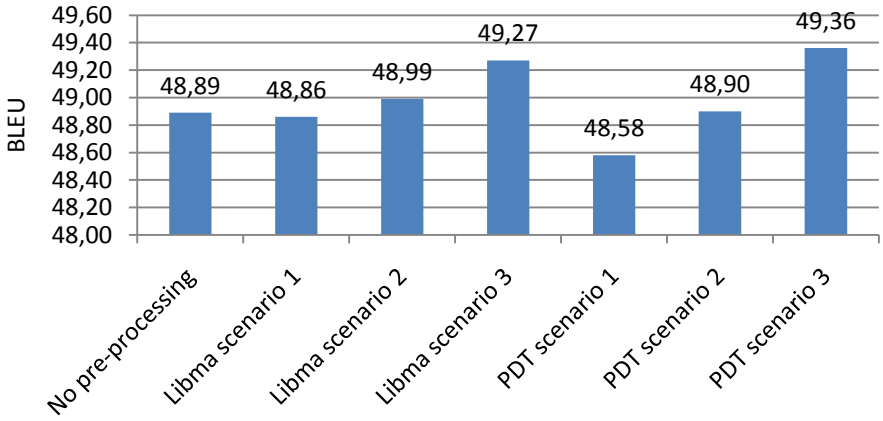


Figure 3.8 Morphological pre-processing of the Acquis corpus[64]

[64] The relevant working directories can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_acquis_[libma|pdt][1|2|3]/

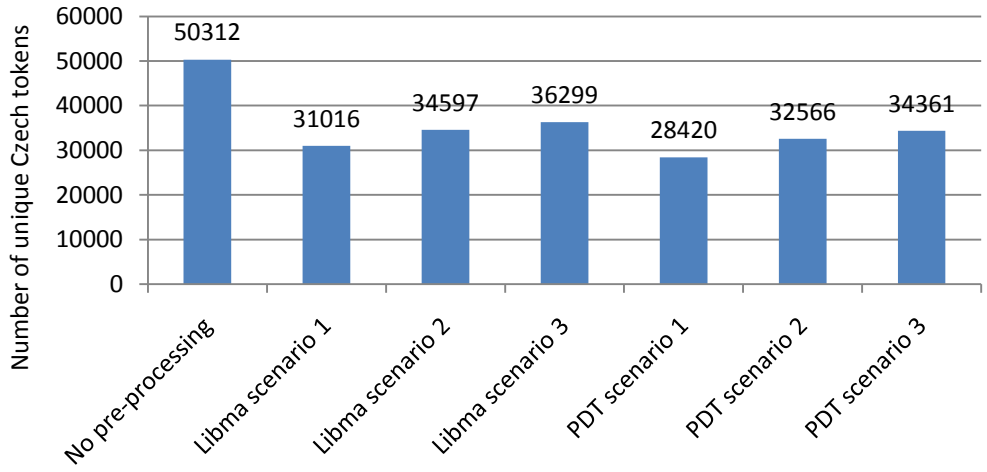


Figure 3.9 Number of unique Czech tokens in the training set of Acquis after morphological pre-processing

From the results we can see that they are not dissimilar from the Kacenka's results. The only thing to observe is that PDT performed better for the third scenario. All in all, the best BLEU increase is about 0.5 % for both Kacenka and Acquis.

Now the question is whether morphological pre-processing would have greater impact if we had fewer training data (smaller corpus). We repeated all the six scenarios for both Kacenka and Acquis but this time taking only 10 % and 50 % of the respective corpus. Let's see the results (Figure 3.10 and Figure 3.11).

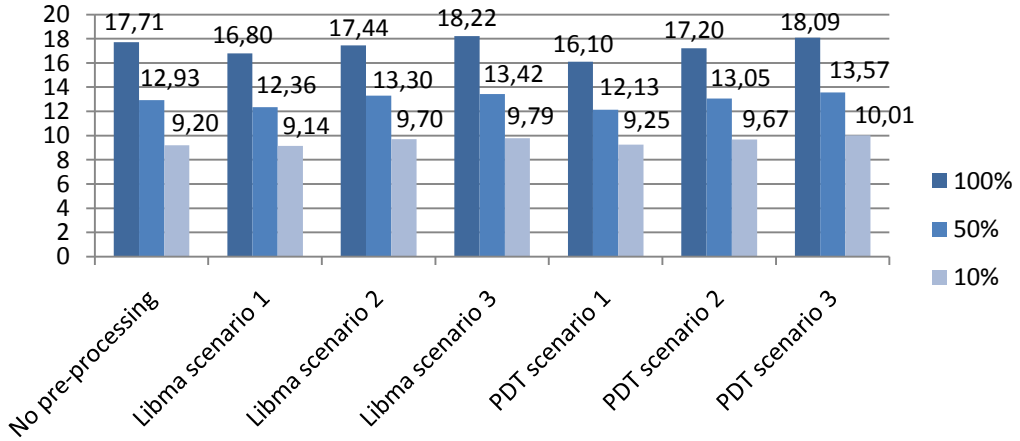


Figure 3.10 Morphological pre-processing of parts of the Kacenka corpus[65]

[65] The relevant working directories can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_kacenka_part_[10|50]_[libma|pdt][1|2|3]/

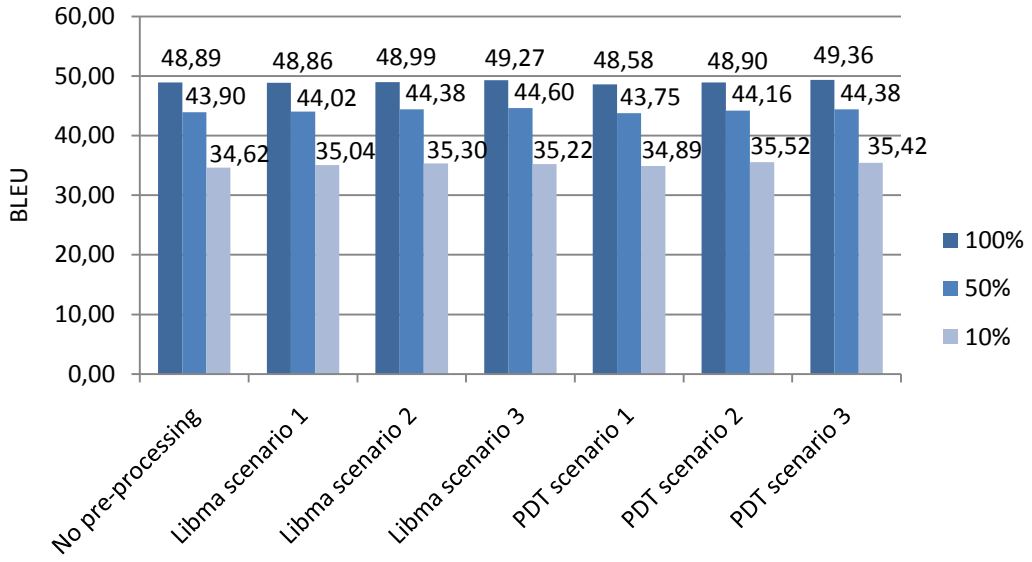


Figure 3.11 Morphological pre-processing of parts of the Acquis corpus[66]

Looking at the graphs, the correlation between corpus size and the effectiveness of morphological pre-processing may not be obvious at first glance. Let's look at the corresponding tables (Table 3.3 and Table 3.4).

Scenario / corpus part	100%	50%	10%
Sentences	102944	51422	10290
No pre-processing	**17,71**	**12,93**	**9,20**
Libma scenario 1	16,80	12,36	9,14
Libma scenario 2	17,44	13,30	9,70
Libma scenario 3	**18,22**	13,42	9,79
PDT scenario 1	16,10	12,13	9,25
PDT scenario 2	17,20	13,05	9,67
PDT scenario 3	18,09	**13,57**	**10,01**
Max increase (% BLEU)	0,51	0,64	0,81

Table 3.3 Morphological pre-processing of parts of the Kacenka corpus

[66] The relevant working directories can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_acquis_part_[10|50]_[libma|pdt][1|2|3]/

Scenario / corpus part	100%	50%	10%
Sentences	150770	76192	14816
No pre-processing	**48,89**	**43,90**	**34,62**
Libma scenario 1	48,86	44,02	35,04
Libma scenario 2	48,99	44,38	35,30
Libma scenario 3	49,27	**44,60**	35,22
PDT scenario 1	48,58	43,75	34,89
PDT scenario 2	48,90	44,16	**35,52**
PDT scenario 3	**49,36**	44,38	35,42
Max increase (% BLEU)	0,47	0,70	0,90

Table 3.4 Morphological pre-processing of parts of the Acquis corpus

Indeed, we can see that corpus size is in indirect proportion to the significance of morphological pre-processing. Moreover, we see that Acquis benefits a bit more from the pre-processing (if we subtract the effects of the fact that Acquis has more sentences).

3.1.4 Combination of individual factors

Now that we have researched several factors that increase the BLEU score, we proceed to build a system that combines the best factors together. We will take the Kacenka corpus along with a dictionary, pre-process it with Libma according to scenario 3 and use it to train the translation model. Then we take the combination of Kacenka + Books 2 corpora to train the language model. Let's see the final BLEU in Figure 3.12.

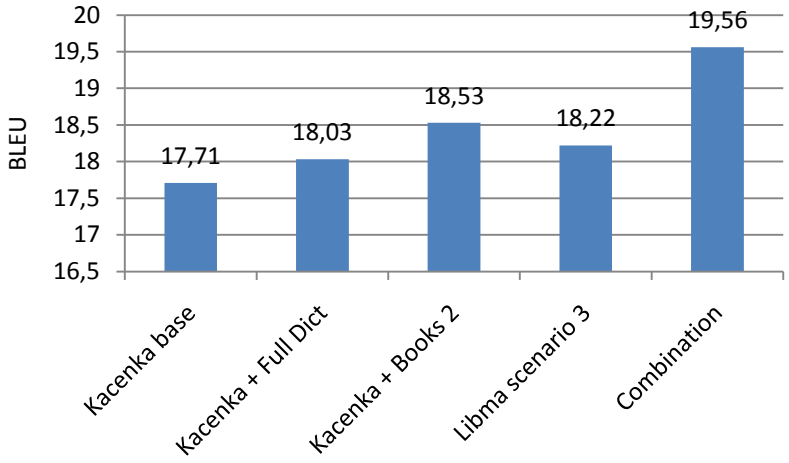


Figure 3.12 Final system combination[67]

We raised the BLEU from 17.71 (base system) to 19.56 (system combination).

[67] The relevant working directory for the combination system can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/work_kacenka_best/

Let's now see the most important thing – the actual translations! After all, BLEU is only an automatic tool to give as an idea of the system performance. But do the sentences read fluently and are they of any use at all? See Table 3.5.

	Input Czech sentence	Output English sentence	Reference English sentence
1	Na to bych moc nespoléhal.	He was not relying on it.	Shouldn't build on it if I were you.
2	Pablo nás tu nechal hnít v nečinnosti.	Pablo left us rot in idleness.	Pablo has rotted us here with inaction.
3	Přišel Anselmo se sekyrou.	Anselmo came the ax.	Anselmo came up with the ax.
4	Neměl se však k odchodu, zřejmě aby si mohl prohlédnout Dixonovu podlitinu na oku.	But he had to leave, apparently to inspect the Dixonovu haematoma in his eye.	He lingered, no doubt to examine Dixon's black eye.
5	Chceš ještě nějaký větve?	You want some branch?	Do you wish more branches?
6	Pravil Dixon odmítavě.	Dixon said disapprovingly.	Thanks, Dixon said dismissively.
7	Zeptal se.	He asked.	He asked.
8	Mně se zdá ten kulomet dobře schovaný.	It seems that gun well hidden.	To me it seems well hidden.
9	Vyjeli na dlouhý úsek rovné silnice, svažující se uprostřed do mělkého dolíku, takže každý metr byl dokonale přehledný.	He rode straight on a long stretch of the road, sloping cloud-roof in a shallow depression, so that every stere was perfectly clear.	They entered a long stretch of straight road, with a slight dip in the middle so that every yard of its empty surface was visible.
10	Větve už ne, řekl Robert Jordan.	And no more, Robert Jordan said.	Not branches, Robert Jordan said.

Table 3.5 Example translations from best system combination (Kacenka)[68]

At first glance you see that the system really works and most of the sentences do read fluently and give (some) sense. However, a close inspection reveals that in some cases the meaning changes, even if the mistake is minor.

Take the first sentence right away. The meaning shifts quite considerably. The original is about person A giving advice to person B, reflecting the situation of person B. The translation shifts the meaning so that a reader could, without a context, assume that person B has already acquiesced in the attitude of person A or that person B never actually relied on *it*. Now consider for example an automatic web translation. If the reader had no knowledge of the original language and did not see the original sentence, he could infer incorrect conclusions about the interaction between person A and person B.

The second sentence, on the other hand, is an almost perfect translation (it only omitted *here*, which could probably be inferred from the context). The third sentence is grammatically

[68] The sentences come from the test set of Kacenka. The recasing and detokenization have been done automatically with the recase.perl and detokenizer.perl scripts. The recaser has been trained with the train-recaser.perl script. All the listed scripts are part of the Moses system package. Finally, the recasing and detokenization has been manually corrected. This was done to improve readability. After all, the original Kacenka corpus is lowercased so the recaser could not be properly trained. The actual words have naturally been left intact.

incorrect but we could suppose the reader could easily infer the *with* conjunction. The fourth sentence is similar to the first sentence. It completely negates the first proposition.

The following sentences resemble the cases already discussed.

3.1.5 Final notes

Despite all the imperfections of the final system we do think that the system could possibly be used as an aid for a professional translator to make his/her work more productive. However, in case of an automatic translation (like web translation) we should be aware that it can lead to quite severe meaning shifts.

Reflecting the final translations, the next iteration of our system's improvement should definitively start with morphological analysis of the negation in Czech verbs and an isolation of the *ne (not)* token out of the verb. Maybe the system would then correctly translate sentences 1 and 4.

You may ask: Why didn't we carry out tuning on the final system? This is because we attempted the tuning both with the mert-moses.pl and the mert-moses-new.pl scripts but the BLEU actually dropped for the test set in both cases (from 17.71 to 16.97 and 17.06, respectively). We could not find a clue as to why this happens so we did not tune the data.

Furthermore, it may be objected that we insufficiently played with various settings of the tools used for training (thresholds for GIZA++, n-gram order of SRILM, decoder weights from moses.ini etc.). That is true. Nevertheless, we had to choose certain limits and trade-offs when deciding upon the contents of this thesis and given the fact that most of the tools are not analyzed down to the implementation level in the theoretical part of the work, this task would eventually require much more space and time in order to be carried out properly.

The final question is: Can the morphological pre-processing possible compensate for an inadequately small training corpus? From our experiments it follows that such a simple pre-processing which we have done is quite ineffective. We get better results simply by feeding the language or translation model with a bit more parallel data (be it Books 2 or another fiction corpus).

3.2 Training for WMT 10

In the final part of this thesis, we attempt the Translation task from the ACL 2010 Joint Fifth Workshop on Statistical Machine Translation (European Commission, 2010a) and compare the results to the best system from Euromatrix Viewing Matrix (Euromatrix Project, 2010c).

We start with the training portion of the CzEng 0.9 corpus, train the models and test the system on the test data from WMT 10 workshop. Next, we repeat the scenario but this time, we concatenate the CzEng 0.9 corpus training data with the training data from WMT 10 (we denote this corpus as WMT 10). Thirdly, we add morphological pre-processing to the second scenario.

We use the Libma library (third scenario – lemmatization, adding pseudo words; leaving frequent word forms intact). We must be careful when determining which words are to be marked as frequent. We decided to extract frequent words only from a combination of the training and developing set of the WMT 10 corpus.

Figure 3.13 shows the results. The last column corresponds to the best system from the Euromatrix Viewing matrix for WMT 10 based on the Moses system (named CU *Moses CS->EN WMT10*). Now actually, the very best system based on BLEU is the *Google CS->EN* system with BLEU 23.4 but here we primarily want to do the comparison of the systems based on Moses[69].

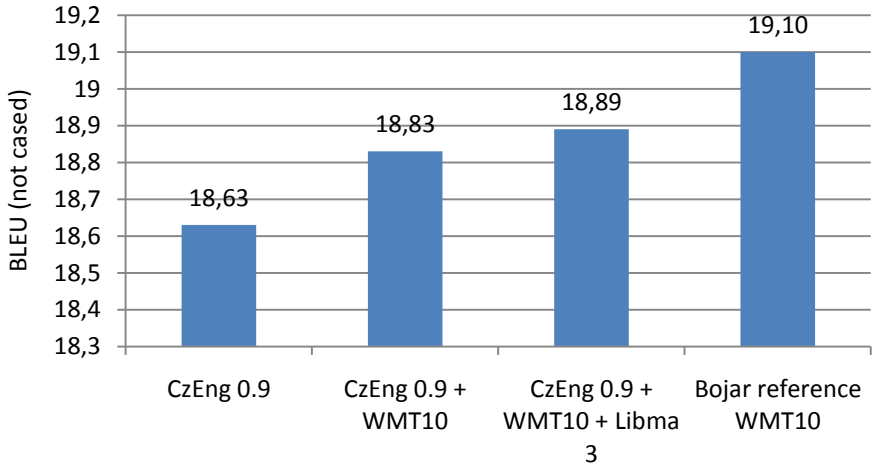


Figure 3.13 Individual scenarios of the WMT 10 Translation Task and their BLEU[70]

As you can see, our system loses 0.21 % BLEU to the *Moses CS->EN WMT10* system.

Finally, we will look the Example translations (Table 3.6).

	Input Czech sentence	Output English sentence	Reference English sentence
1	Barack Obama dostane jako čtvrtý americký prezident Nobelovu cenu míru	Barack Obama gets as the American President the Nobel Peace Prize	Barack Obama becomes the fourth American president to receive the Nobel Peace Prize
2	Americký prezident Barack Obama přiletí do norského Osla na 26 hodin, aby si zde jako čtvrtý americký prezident v historii převzal Nobelovu cenu míru.	President Barack Obama will arrive in Oslo, Norway on 26 hours to get here as the American president in history took the Nobel Peace Prize.	The American president Barack Obama will fly into Oslo, Norway for 26 hours to receive the Nobel Peace Prize, the fourth American president in history to do so.
3	Diplom, medaili a šek na 1,4 milionů dolarů dostane mimo	The diploma, medals and a check for $1.4 million gets in efforts to	He will receive a diploma, medal and cheque for 1.4 million

[69] Another imperfection to note is that our system has been evaluated with the multi-bleu.perl script while the WMT10 systems are evaluated with the mteval-v11b.pl script. However, both these scripts compute a standard BLEU score on lowercased data.

[70] The work can be found at minerva1:/mnt/minerva1/nlp/projects/mt/work/moses_tests/ in the following directories: work_czeng2_base, work_czeng2_news_base, and work_czeng2_news_libma3.

	jiné za výjimečné úsilí o posílení světové diplomacie a spolupráce mezi národy.	strengthen the world diplomacy and cooperation among nations.	dollars for his exceptional efforts to improve global diplomacy and encourage international cooperation, amongst other things.
4	Šéf Bílého domu přiletí do norské metropole ráno i s manželkou Michelle a bude mít napilno.	The chief of the White House to Norwegian metropolis morning with his wife, Michelle, and will have been busy.	The head of the White House will be flying into the Norwegian city in the morning with his wife Michelle and will have a busy schedule.
5	Nejprve zavítá do Nobelova institutu, kde se vůbec poprvé setká s pěti členy výboru, který ho v říjnu vybrali ze 172 lidí a 33 organizací.	First come to the Institute, where the first meets with five members of the Committee, in October the 172 people and 33 organisations.	First, he will visit the Nobel Institute, where he will have his first meeting with the five committee members who selected him from 172 people and 33 organisations.

Table 3.6 Example translations of our system (WMT 10)

As you can see, the translation completely misses the *fourth* numeral in the first and second sentence. Another problem may be an unclear specification of subject (due to the pronoun pro-drop nature of Czech). However, we could assume that the reader would in some cases be able to infer the subject from context.

3.2.1 Final notes

Despite the abovementioned imperfections, the output sentences are quite readable and faithful to the original. We suppose this is mainly due to the considerable size of the CzEng 0.9 corpus. We saw that our morphological pre-processing had only minor effects on the BLEU.

We did not outperform the reference system from EuroMatrix Viewing Matrix. However, we did come quite close and we gained valuable working experience with state-of-the-art statistical machine translation.

Conclusion

This Master's thesis elaborated both on **theory and practical issues** concerning building a statistical machine translation system. The cornerstone and the main deliverable of the thesis is its empirical part. Firstly, we thoroughly analyzed the time requirements of multithreaded modifications of GIZA++ word alignment tools. Secondly, we empirically analyzed several factors that influence the quality of the translations of the SMT system.

We showed that by using **MGIZA++** we can reduce the time needed to perform word alignment down to about 20 % (depending on the corpus characteristics; compared to standard GIZA++). Furthermore, we found out that running **PGIZA++** on a cluster may exceed the performance of MGIZA++ but only if the corpus is large. We also learnt that using PGIZA++ is more complicated than using MGIZA++.

As to the **quality of the translations**, we determined that the relation between corpus size and the resulting BLEU is roughly linear. We found out that incorporating extra bilingual data from the same domain into the language model improved BLEU quite considerably. We showed that including a bilingual dictionary or doing a morphological pre-processing on the Czech input can slightly increase BLEU. We also saw that the effects of morphological pre-processing are in indirect proportion to corpus size, and that the simplest pre-processing (pure lemmatization) can in fact decrease BLEU.

Finally, we demonstrated that our system's performance is comparable to the performance of best systems in the **Euromatrix Viewing Matrix**.

After inspecting the final translations, we suggested that a possible **future improvement** of the system should focus on a more sophisticated morphological pre-processing of Czech verbs. The rationale for this is to eliminate certain mistakes that could lead to misinterpretation of the translated sentences.

There are also other possible ways of improving the system. We could play with various parameters of the tools (GIZA, SRILM, Moses decoder). We could also completely abandon Moses and use for example Cunei or Joshua.

In the course of working on the project, several new things happened. At end of 2009, Qin Gao announced that the development of PGIZA++ had been discontinued and that his efforts

concentrate on MGIZA++, which is now integrated with Chaksi and can be run on Hadoop Clusters (Gao, 2009).

Another update comes from the Moses system. One of the additions listed for year 2010 is the so called ***Experimental management system*** whose purpose is actually quite similar to what this thesis dealt with: It should help you doing experiments with Moses and compare its performance under different scenarios.

To sum it up, we saw that machine translation is no science fiction. Computers are able to do translation and can help people gain access to information which they would otherwise not understand. The SMT can also improve the productivity of professional translators. Nevertheless, we must be aware of the limitations of machine translation and especially, of seemingly insignificant meaning shifts that can, in fact, have severe consequences. Despite all the improvements and active research in the field of natural language processing the author believes that professional human translators and interpreters can never be fully substituted by a computer.

References

[1] Bartoň, R. 2009. *Parallel Corpus Joint-Multigram Training 2* [online]. Last revision 24 February 2010 [cited: 18 May 2010]. Available at <https://merlin.fit.vutbr.cz/nlp-wiki/index.php/Parallel_Corpus_Joint-Multigram_Training_2>.

[2] Bojar, O., et al. 2007. *CzEng 0.7 (Czech-English Parallel Corpus, version 0.7)* [online]. Last revision 2008 [cited: 18 May 2010]. Available at <http://ufal.mff.cuni.cz/czeng/czeng07/>.

[3] Bojar, O., et al. 2009a. *CzEng 0.9 (Czech-English Parallel Corpus, version 0.9)* [online]. Last revision 2009 [cited: 18 May 2010]. Available at <http://ufal.mff.cuni.cz/czeng/czeng09/>.

[4] Bojar, O., Žabokrtský Z. 2009b. *CzEng 0.9: Large Parallel Treebank with Rich Annotation*. Prague Bulletin of Mathematical Linguistics, 92.

[5] Callison-Burch, C. 2009. *HOW-TO GUIDE: Installing and running the Joshua Decoder* [online]. Last revision 12 June 2009 [cited: 18 May 2010]. Available at <http://cs.jhu.edu/~ccb/joshua/>.

[6] Cuřín, J., Bojar, O. 2007. *Machine Translation Projects at ÚFAL* [online]. Last revision 22 May 2007 [cited: 18 May 2010]. Available at <http://ufal.mff.cuni.cz/~curin/mt/>.

[7] Euromatrix Project. 2008a. *Moses – Background* [online]. Last revision 21 December 2008 [cited: 18 May 2010]. Available at <http://www.statmt.org/moses/?n=Moses.Background>.

[8] Euromatrix Project. 2008b. *Parallel Corpora Available On-Line* [online]. Last revision 5 August 2008 [cited: 18 May 2010]. Available at <http://www.statmt.org/moses/?n=Moses.LinksToCorpora>.

[9] EuroMatrix Project. 2010a. *The EuroMatrix Project (Sept. 2006 - Febr. 2009)* [online]. Last revision 2010 [cited: 18 May 2010]. Available at <http://www.euromatrix.net/>.

[10] Euromatrix Project. 2010b. *Advanced Features of the Decoder. Cube pruning* [online]. Last revision 4 May 2010 [cited: 18 May 2010]. Available at <http://www.statmt.org/moses/?n=Moses.AdvancedFeatures#ntoc13>.

[11] Euromatrix Project. 2010c. *System Output List* [online]. Last revision 2010 [cited: 19 May 2010]. Available at <http://matrix.statmt.org/matrix/systems_list/1621>.

[12] EuroMatrixPlus Consortium. 2010a. *The EuroMatrixPlus Project* [online]. Last revision 2010 [cited: 18 May 2010]. Available at <http://www.euromatrixplus.net/>.

[13] European Commission. 2009. *The JRC-Acquis Multilingual Parallel Corpus* [online]. Last revision 11 June 2009 [cited: 18 May 2010]. Available at <http://langtech.jrc.it/JRC-Acquis.html>.

[14] European Commission. 2010a. *ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics Matr.* [online]. Last revision 2010 [cited: 18 May 2010]. Available at <http://www.statmt.org/wmt10/index.html>.

[15] European Commission. 2010b. *Shared Task: Machine Translation for European Languages* [online]. Last revision 2010 [cited: 18 May 2010]. Available at <http://www.statmt.org/wmt10/translation-task.html>.

[16] Gao, Q., Vogel, S. 2008. *Parallel Implementations of Word Alignment Tool* [online]. Last revision June 2008 [cited: 18 May 2010]. Available at <http://www.aclweb.org/anthology/W/W08/W08-0509.pdf>.

[17] Gao, Q. 2009. *Welcome to my home page* [online]. Last revision 2010 [cited: 18 May 2010]. Available at <http://www.cs.cmu.edu/~qing/>.

[18] Gao, Q. 2009b. *MGIZA++ Configuration* [online]. Last revision 11 December 2009 [cited: 18 May 2010]. Available at <http://geek.kyloo.net/software/doku.php/mgiza:configure>.

[19] Goldwater, S., McClosky, D. 2005. *Improving Statistical MT through Morphological Analysis. Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing.* Vancouver, 2005.

[20] Google Code. 2009. *GIZA++ statistical translation models toolkit*. *Issue 7: Cannot compile with gcc 4.3 or greater* [online]. Last revision 4 February 2009 [cited: 18 May 2010]. Available at <http://code.google.com/p/giza-pp/issues/detail?id=7>.

[21] Hana, J., Zeman, D. 2005. *Manual for Morphological Annotation* [online]. Last revision 19 May 2005 [cited: 18 May 2010]. Available at <http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/m-layer/pdf/m-man-en.pdf>.

[22] Hunglish Project. 2009. *Hunalign - sentence aligner* [online]. Last revision 21 September 2009 [cited: 18 May 2010]. Available at <http://mokk.bme.hu/resources/hunalign>.

[23] Jurafsky, D., Martin, J. H. 2009. *Speech and language processing*. New Jersey: Prentice Hall, 2009. 1024 p. ISBN 978-0-13-187321-6.

[24] Koehn, P. 2010. *Moses. Statistical Machine Translation System. User Manual and Code Guide* [online]. Last revision: 18 May 2010 [cited: 18 May 2010]. Available at <http://www.statmt.org/moses/manual/manual.pdf>.

[25] Manning, C. D., Schütze, H. 1999. *Foundations of statistical natural language processing*. Cambridge, Massachusetts: The MIT Press, 1999. 620 p. ISBN 978-0262133609.

[26] Och, F. J. 2001. *GIZA++: Training of statistical translation models* [online]. Last revision 30 January 2001 [cited: 18 May 2010]. Available at <http://www.fjoch.com/GIZA++.html>.

[27] Peloušková, H., Káňa, T. 2007. *Tvorba, funkce a využití Česko-německého paralelního korpusu* [online]. Last revision 21 October 2007 [cited: 18 May 2010]. Available at <http://is.muni.cz/publikace/publikace_simple.pl?lang=en;id=726491>.

[28] Phillips, A. B., Brown, R. D. 2009. *Cunei Machine Translation Platform: System Description. 3rd Workshop on Example-Based Machine Translation*. Dublin, Ireland, 2009.

[29] Plitt, M., Masselot, F. 2010. *A productivity Test of Statistical Translation Post-Editing in a Typical Localisation Context.* Prague Bulletin of Mathematical Linguistics, 93.

[30] SRI International. 2009. *SRILM - The SRI Language Modeling Toolkit* [online]. Last revision 9 November 2009 [cited: 18 May 2010]. Available at <http://www-speech.sri.com/projects/srilm/>.

[31] Šlancarová, D. 2003. *The KAČENKA 2 project* [online]. Last revision 2003 [cited: 18 May 2010]. Available at <http://www.phil.muni.cz/angl/kacenka2/>.

[32] Tiedermann, J. 2007. *OpenSubtitles* [online]. Last revision 2007 [cited: 18 May 2010]. Available at <http://urd.let.rug.nl/tiedeman/OPUS/OpenSubtitles.php>.

[33] ÚFAL. 2007. *Institute of Formal and Applied Linguistics* [online]. Last revision 2007 [cited: 18 May 2010]. Available at <http://ufal.mff.cuni.cz/>.

Subject index

List of figures

List of tables

Appendix A – Scripts created

This section lists important python scripts that have been created for this project. In case that the entire script or its part has been inspired by another script found on the Internet, the URL of the source is stated in the script header.

The scripts are stored at minerva1:/mnt/minerva1/nlp/projects/mt/tools/myown/ and on the accompanying CD as well. All the scripts need to be interpreted with Python. Some of them may require being located in a specific directory (due to relative paths). In such case, the script header contains a hint as to which directory the script should be run from.

1 Corpora preparation

Script	Description
czeng.py	Preparation of the CzEng 0.9 corpus.
dict.py	Preparation of Czech-English dictionary.
kacenka.py	Preparation of the Kacenka corpus. Section 2.1.1.1 gives a comprehensive description of this script.
myclean.py	This script cleans the input corpus from unnecessary quotation marks and apostrophes.
unwrap-xml.py	This script eliminates SGML formatting from the input corpus and returns plain text on output.

2 GIZA++ training

Script	Description
preparegiza.py	Copies supporting files (*.vcb, *.classes, *.cats etc.) for GIZA++ run from one location to another.
traingiza.py	Trains a corpus with MGIZA++.

3 Moses training

Script	Description
frequent_words.py	Prints a list of words that occur in the input corpus with a frequency exceeding the given frequency limit.
morphology_ma.py	Does a morphological pre-processing on the input corpus using the Libma library.
morphology_ma_env.py	This script is an envelope for the morphology_ma.py script and should be run in case the morphology.py scripts uses too much memory, gets a SIGABRT and cannot finish the work on large corpora.
morphology_pdt.py	Does a morphological pre-processing on the input corpus using the Prague Dependency Treebank.
moses.py	This is the main script used for testing and tuning various modifications of the Moses system.

Appendix B – Working directories

The root directory of the thesis is minerva1:/mnt/minerva1/nlp/projects/mt/. It contains these subdirectories:

…/mt/	Description
/corpora/	This directory contains all corpora. The names of the subdirectories correspond to names of the corpora. Some of the directories contain combinations of two corpora. Refer to section 2.3.2.1 on how the combination corpora have been created.
/tools/	This directory contains both sources and executables of third-party tools used throughout the work.
/work/	In this directory, all work has been accomplished.
/work/czeng_preprocessing/	Preparation of the CzEng 0.9 corpus (see section 2.1.5.1)
/work/dict_preprocessing/	Preparation of Czech-English dictionary (see section 2.1.7)
/work/ kacenka_preprocessing/	Preparation of the Kacenka corpus (see section 2.1.1.1)
/work/mgiza_tests/	Performance tests of MGIZA++ (see section 2.2.1)
/work/moses_tests/	Tests of the Moses system (see chapter 3)
/work/pgiza_tests/	Performance tests of PGIZA++ (see section 2.2.2)

Appendix C – Corpora statistics

	Sentences	Czech tokens	English tokens	Average sentence	Total size (UTF-8)
Acquis	234 320	5 804 785	6 752 251	26,8 words	71 MB
Books 2	982 937	12 467 864	14 602 134	13,8 words	133 MB
CzEng 0.7	1 096 940	15 292 171	17 868 659	15,2 words	184 MB
CzEng 0.9	8 029 801	80 256 429	92 522 247	10,8 words	890 MB
Kacenka 2	118 285	1 523 903	1 697 637	13,7 words	17 MB
Open Subtitles	377 623	2 458 480	3 086 874	7,4 words	25 MB
WMT10	99 756	2 171 419	2 378 823	22,9 words	27 MB
Lite Dict	80 960	93 948	99 292	1,2 words	2 MB
Full Dict	293 020	515 576	597 341	1,9 words	8 MB

All corpora are stored in the minerva1:/mnt/minerva1/nlp/projects/mt/corpora/ directory.

The token counts are measured after tokenization.