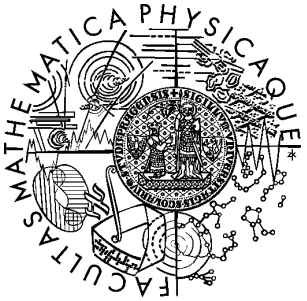
Charles University in Prague
Faculty of Mathematics and Physics

**MASTER'S THESIS**




Kamil Kos

# Rich Features in Phrase-Based Machine Translation

Institute of Formal and Applied Linguistics

Supervisor: RNDr. Ondřej Bojar, Ph.D.

Study program: Computer Science, Mathematical Linguistics

I would like to express my gratitude to my supervisor, RNDr. Ondřej Bojar, Ph.D, for his valuable suggestions and consultation time. My sincere thanks also belong to all authors of NLP tools and resources that I used in this thesis.

I hereby declare that I have prepared this thesis on my own, using only the materials cited. Ideas taken from other sources are identified as such. I give consent to publish this thesis.

December 21, 2010 Kamil Kos

# Contents

Název práce: Bohaté rysy ve frázovém strojovém překladu
Autor: Kamil Kos
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí diplomové práce: RNDr. Ondřej Bojar, Ph.D.
e-mail vedoucího: bojar@ufal.mff.cuni.cz
Klíčová slova: strojový překlad, hodnocení kvality, kontextový model, suffixové pole
Abstrakt:

V této práci zkoumáme metody, jak zlepšit kvalitu statistického strojového překladu použitím bohaté lingvistické informace. Nejdříve popíšeme SemPOS – metriku, která využívá mělké sémantické reprezentace vět k hodnocení kvality strojového překladu. Ukážeme, že i když tato metrika dosahuje vysoké korelace s lidskými hodnoceními kvality překladu, není samostatně vhodná pro optimalizaci parametrů systémů strojového překladu. Za druhé rozšíříme základní log-lineární model používaný ve statistickém strojovém překladu o kontextový model zdrojové věty, který pomáhá lépe rozlišovat mezi různými možnostmi překladu dané fráze a pomáhá vybrat nejvhodnější překlad pro daný kontext v aktuální větě.

Title: Rich Features in Phrase-Based Machine Translation
Author: Kamil Kos
Department: Institute of Formal and Applied Linguistics
Supervisor: RNDr. Ondřej Bojar, Ph.D.
Supervisor's e-mail address: bojar@ufal.mff.cuni.cz
Keywords: machine translation, quality evaluation, source-context model, suffix array
Abstract:

In this thesis we investigate several methods how to improve the quality of statistical machine translation (MT) by using linguistically rich information. First, we describe SemPOS, a metric that uses shallow semantic representation of sentences to evaluate the translation quality. We show that even though this metric has high correlation with human assessment of translation quality it is not directly suitable for system parameter optimization. Second, we extend the log-linear model used in statistical MT by additional source-context model that helps to better distinguish among possible translation options and select the most promising translation for a given context.

# Chapter 1

# Introduction

Machine translation (MT) uses computers for automatic translation between natural languages. So far translation quality of state-of-the-art MT systems has not reached the level which is required for high-quality translation of texts on general topic, which would give us the possibility to use this technology without human supervision in large scale. Therefore, new methods need to be investigated and existing models need to be extended to provide automatic translation of better quality. In this thesis we focus on improving existing methods used in statistical MT systems to provide better translation quality by employing linguistically rich information.

## 1.1 Goal of Machine Translation

At present time, MT is still not capable of preserving correct meaning of complex sentences. It can happen that the meaning of the translated text differs from the original because negation is used incorrectly, another meaning of a word is used, or the translated sentence just does not make any sense. Therefore, MT is usually used for gist translation, where we do not need to understand the exact meaning of a text but rather prefer fast orientation in foreign language texts, or as a support tools in word processors to facilitate translation done by humans. For high quality translation we still need to seek human translators who are able to guarantee that the meaning of a text is preserved.

To get a better notion of the current state of machine translation we present a short example of MT system output in Figure 1.1, which was obtained by translating the Czech Wikipedia entry about machine translation[1] by a statistical MT system. The translated text contains a lot of the information from the original article in Czech, but many sentences are ungrammatical or difficult to understand without previous knowledge about the domain.

[1] http://cs.wikipedia.org/wiki/Strojový_překlad

Machine translation (angl. machine to) is a process of an automatic Translation from one natural language to another using computers. Machine translation is a long time an important role of computer science, however, is not completely satisfactorily resolved still. Today many systems are available, whose output is not perfect, but it is sufficient quality for use in many areas where helps human translators.

Figure 1.1: Example of a state-of-the-art MT system output.

There are several factors that influence the quality of MT in general. The most important factor is the size of the domain in which we try to translate. If we select a restricted domain with clearly defined vocabulary, the translation task is relatively easy. The translation system Météo (Chandioux and Guraud; 1981) is a good example. This system was developed in Canada in the 1980's to translate weather forecasts between French and English and it was successfully used for almost 20 years. Since the domain covered only weather terminology and the translated text had a very rigid structure, the system could be relatively simple and it was possible to operate it even with hardware available at that time.

| *Bank* | | |
|---|---|---|
| | Meaning | Czech translation |
| 1. | business that provides financial services | *banka* |
| 2. | land along the side of a river | *břeh* |
| 3. | a large pile of earth, sand, snow | *násep* |
| 4. | money in a gambling game that people can win | *bank* |

Figure 1.2: Possible meanings of the English noun *bank* with different Czech translation.

If we extend the translation domain, or even do not restrict it at all, we need to deal with ambiguity which is innate to every natural language in form of synonyms[2] and homonyms.[3] Homonyms represent a serious issue because the word alone does not give us any clue which meaning (and the corresponding translation) is correct. Figure 1.2 shows some possible meanings of the English noun *bank*.[4] There are four meanings of the word that are expressed by a different translation in Czech. The most probable translation would be the first one (*banka* - financial institution) but there are situations where one of the other meanings should be used, e.g. the translation *břeh* in the phrase *river bank*. If we select the wrong meaning the translated text can be very difficult

[2]Two words are synonyms if they have the same meaning, e.g. *buy* and *purchase*.
[3]Homonyms are words with the same form but different meaning, e.g. *bank*.
[4]This list is not exhaustive - *bank* can also have other meanings.

to understand. Therefore, it is necessary to look for hints in the text which meaning could be the correct one. We can infer the correct meaning of the word being translated for example from the topic of the text or words co-occurring in the sentence. However, this makes the system more complex because the translation of one word is dependent on the context in which it occurs. Thus, it is necessary to design MT systems with good balance between the complexity, computational viability of the underlying model and available data.

Another factor that strongly affects the quality of machine translation is the selection of the source and target language. It is easier to translate between languages that have similar sentence structure and grammar rules, e.g. Spanish and French, than between languages that belong to different language families, e.g. Czech and English or Greek and Finnish.[5] In the former case we can often translate sentences word by word using only a simple dictionary without any reordering of words, while it is necessary to perform complex transformations of the sentence structure in the latter one. If computers need to significantly restructure a sentence because the target language has completely different grammar rules than the source languages, a lot of small mistakes can occur on the way. The more difficult the target language is, the more mistakes can happen. This applies especially to languages with rich morphology, e.g. Czech. When translating from English to Czech, the structure of the sentence can be more or less preserved but a lot of additional information has to be provided in the target language because one word can take many different forms depending on number, gender or other grammatical categories. If we assign one of these categories wrong, the resulting word form can be different and this one mistake can change the meaning of the whole sentence.

Current MT research aims at developing systems that are capable of translating text on an arbitrary topic and between two arbitrary languages. This is a big challenge because the underlying model needs to be sufficiently flexible in order to accommodate also language-dependent phenomena.

## 1.2 Current MT Development

The prevailing research direction in MT has been represented by *statistical MT* in recent years. This term covers a wide range of theoretical approaches to MT which have one fundamental feature in common: they use large amounts of mono- and bilingual texts to *automatically* learn rules to translate from one language to another language. Therefore they are relatively easy and fast to deploy because they do not require any special knowledge about the languages in consideration given the mono- and bilingual texts.

[5] The site `http://www.statmt.org/matrix/` shows the translation complexity (expressed as BLEU score) for selected European language pairs.

There are two ways to improve the performance of a statistical MT system. First, it is possible to produce even larger collections of mono- and bilingual texts to get better coverage of the languages and estimate the translation rules more accurately. This approach is easy to do but it requires a lot of time and human effort to find and select suitable sources of bilingual training data. Usually, it is easy to find parallel texts for a specific domain, e.g. legal documents produced in states or organizations that have multiple official languages. However, for some other domains like newspaper articles that cover general topics it is difficult to find a suitable source of data because they are usually produced only in one language.

The second approach to improve the quality of statistical MT systems is to enhance the underlying model by additional features that help to generate better translation from available data. This can be achieved by deeper analysis of the data and learning additional translation rules from it. Although this makes the model more complex, it is a promising approach for the near future because hardware is becoming faster every year and current statistical MT models can be easily run in parallel environment since they usually translate individual sentences separately. Thus, incorporating additional sources of information can bring the desired quality improvement while keeping MT systems suitable even for real-time applications. Harnessing syntactic and semantic representation of a sentence seems to have high potential for future even though suitable models taking advantage of this information are still to be developed.

An important step to better MT output is a translation quality metric that can recognize a good translation from a bad one. In recent years new metrics were proposed using rich annotation of text, e.g. part-of-speech tags. These metrics show better correlation to human assessment of translation quality than metrics working only on surface forms. However, the required annotation of the evaluated text makes the metrics language dependent and they must be adapted for each language.

## 1.3 Outline

In this thesis we focus on improving statistical MT output quality by using rich annotation of data. We use the phrase-based statistical MT system Moses (Koehn et al.; 2007) as our baseline system. We investigate the following two methods:

1. First, we try to employ an MT quality metric which uses rich annotation of the evaluated text to optimize model parameters of the MT system. The metric was designed directly for Czech and showed better correlation results with human assessment of translation quality for Czech than the metric commonly used for parameter optimization.

2. Second, we design an extension of the phrase-based log-linear model. This extension takes

advantage of rich source-side annotation to generate better translation. The extension is based on suffix arrays, an efficient data structure that allows to compute feature values on-demand during the translation process with low time costs.

We structure our work in the following way: first, we introduce the field of machine translation and related concepts in Section 2. In Section 3, we investigate the possibility of improving MT system performance by optimizing parameters of the underlying model by employing a more suitable MT quality metric. In Section 4, we describe an MT model extension which uses context information from the source sentence to improve translation quality. Then, we discuss the obtained results and comment on their contribution to better translation in Section 5. Finally, we conclude in Section 6.

# Chapter 2

# Background

Before we start with the description of our work, we introduce the field of machine translation. We focus especially on statistical MT, which represents one of the fastest developing streams in MT and which has been receiving a lot of research attention in recent years. Thanks to this fact, statistical MT has been able to report steady improvement in translation quality in the last decade and is capable of translating even long sentences relatively accurately. However, flawless automatic translation between natural languages is still an issue for future research.

## 2.1 Machine Translation

The first attempts to use computers for automatic translation between two natural languages started shortly after the design of the first computers in the 50's of the $20^{th}$ century. One of the first experiments with automatic translation dates back to 1954 when IBM and Georgetown University tried to translate from Russian to English (Hutchins; 1954). A manually created rule-based grammar and vocabulary restricted to only 250 words made it possible to translate short sentences accurately. A lot of expectations were raised by this project, but nearly half of the century was required to arrive at a point at which MT could be used for translation of an arbitrary sentence.

Various approaches to MT have been proposed in recent years ranging from simple human-written translation rules to linguistically motivated complex translation systems. As more computational power became available, statistical approaches gained on significance. Their biggest advantage lies in their universal applicability and fast deployment. Within a few days or even hours, it is possible to train a translation system for two arbitrary languages. The only requirement is a large collection of text in both languages that are aligned sentence by sentence. Obtaining such large collections of text, called corpora, is becoming easier since there are vast amounts of text on the Internet that can be used for training purposes after careful extraction.

## 2.2 Statistical Machine Translation

Statistical MT was pioneered by IBM researchers (Brown et al.; 1990), who based their work on the *noisy-channel* model. The idea of using the noisy-channel model dates back to the 1950's when it was introduced in the information theory to retrieve the original message from data sent over an unreliable and noisy transmission channel. However, it took several decades until the noisy-channel model could be applied in machine translation because computers were not fast enough and machine-readable corpora of bilingual text were difficult to obtain. In the early 1990's, both resources were available in sufficient amount. Since then, statistical machine translation evolved quite rapidly.

### 2.2.1 Noisy-Channel Model

The idea of the noisy-channel model is simple. Imagine that we want to translate from English sentence $f$ to Czech sentence $e$. We want to find the Czech sentence $\hat{e}$ that maximizes the posterior probability $P(e|f)$ given the English sentence $f$ (see Equation 2.1). We can apply the Bayes' Theorem and receive the following equations:

$$\hat{e} = \underset{e}{argmax}\, P(e|f) \tag{2.1}$$

$$= \underset{e}{argmax}\, \frac{P(f|e) \times P(e)}{P(f)} \tag{2.2}$$

$$= \underset{e}{argmax}\, P(f|e) \times P(e) \tag{2.3}$$

where $P(f|e)$ is the probability of translating the English sentence $f$ to the Czech sentence $e$ and $P(e)$ is the probability of the Czech sentence $e$.[1] In Equation 2.3 it is possible to omit $P(f)$ because the English sentence $f$ is given and it is constant for all Czech sentences over which we maximize. Therefore, it does not have any influence on the maximization step and we can take it out.

The two basic constituents of the noisy-channel model are the probabilities $P(f|e)$ and $P(e)$. Probability $P(f|e)$ is called *translation model* because it models the probability that a Czech sentence $e$ is translated into an English sentence $f$, and probability $P(e)$ is called *language model* because is models the probability of a sentence in a given language.

The noisy-channel model is depicted in Figure 2.1. Informally, we can imagine someone thinking in Czech but saying everything in English. The produced English sentence can be viewed

[1] We use letters $f$ and $e$ to denote the source and the target sentence due to historical convention, which was introduced in publications about the *F*rench to *E*nglish statistical translation system Candide. The $f$ can be also viewed as an acronym for *foreign* language.

as scrambled Czech sentence that was originally in his or her mind. However, we would like to know what the person was originally thinking and not what he or she said. Therefore, the translation task is to restore the unscrambled sentence in Czech from the corrupted English sentence.

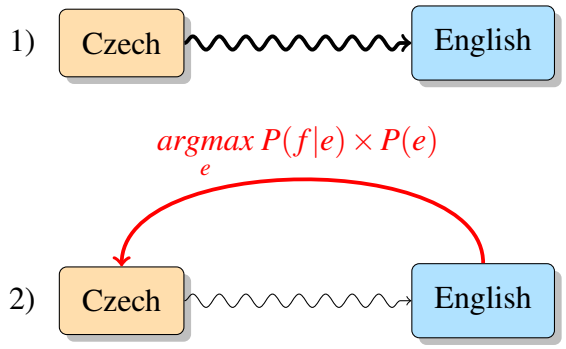



Figure 2.1: The noisy-channel model combines the language model $P(e)$ and the translation model $P(f|e)$ to find the best Czech translation of an English sentence $f$.

### Language Model

Now, we will briefly describe the language model. As already mentioned, the language model is represented by the probability $P(e)$ which is the probability of the sentence $e$ in a given language. This probability can be expressed as

$$P(e_1 \dots e_l) = P(e_1) \cdot P(e_2|e_1) \cdot P(e_l|e_1 e_2 \dots e_{l-1}) \tag{2.4}$$

where $e = e_1 \dots e_l$ is the target language sentence of the length $l$ and $P(e_i|e_1 e_2 \dots e_{i-1})$ the conditional probability that word $e_i$ follows the sequence of words $e_1 e_2 \dots e_{i-1}$. However, it is not possible to store the conditional probability for all possible sequences of words because of their large number due to exponential explosion. Therefore, we use an approximation instead, which is based on n-grams. N-gram is a sequence of $n$ words. For a language model of order $n$ we use the approximation

$$P(e_i|e_1 \dots e_{i-1}) \approx P(e_i|e_{i-n+1} \dots e_{i-1}). \tag{2.5}$$

This means that we reduce the history we need to remember for every word only to $n-1$ preceding words.

Formally, a language model of order $n$ can be expressed as

$$P(e_1 \dots e_l) = \prod_{i=1}^{l} P(e_i|e_{i-n+1} \dots e_{i-1}) \tag{2.6}$$

where $e = e_1 \dots e_l$ is a sentence of the length $l$ and $P(e_i|e_{i-n+1} \dots e_{i-1})$ the conditional probability that word $e_i$ follows the sequence of words $e_{i-n+1} \dots e_{i-1}$.

There is a major problem connected with this approach. It can happen that the probability $P(e_i|e_{i-n+1}\dots e_{i-1})$ is zero because we have not seen the sequence $e_{i-n+1}\dots e_i$ in the training data. This would mean that the whole sentence would get zero probability. To overcome this problem it is necessary to apply smoothing techniques that use shorter history up to uniform distribution to back-off the n-gram language model.

#### Translation Model

The translation model is responsible for generating possible translation options for source sentence fragments. Current state-of-the-art translation models follow one of the two basic approaches:

- phrase-based,
- hierarchical (syntax-based) model.

Phrase-based translation models are simpler because they try to create sentence translations by concatenating translations of source word sequences, called *phrases*. Phrases are taken as basic units and are translated at once. On the other hand, hierarchical models work with the syntactical tree of the sentence.

Hierarchical models represent a promising branch of statistical MT because they can better handle structural differences in the source and target language. This is due to the fact that they explicitly work with syntactic dependencies among the words in form of a parse tree. Nevertheless, their performance is still inferior to the state-of-the-art phrase-based statistical MT systems (Callison-Burch et al.; 2008, 2009). Several formalisms for hierarchical translation model were proposed using parse tree either on the target side, e.g. Yamada and Knight (2001); Gildea (2003), both sides, e.g. Shieber and Schabes (1990); Gildea (2003); Eisner (2003), or neither of the sides, inducing it from parallel data Chiang (2005). It is out of scope of this thesis to describe them more in detail. We cover only the phrase-based translation model in Section 2.2.3 since our source-context model is an extension of the phrase-based model.

### 2.2.2 Log-Linear Model

The most wide-spread model that is currently used in statistical MT is the log-linear model (Och and Ney; 2002). It is a generalization of the noisy-channel model because it allows to add also other components, typically called *features*, beside the language and the translation model.

The log-linear model uses a set of feature functions $h_n(e,f)$ that take the source sentence $f$ and the target sentence $e$ as parameters. Each feature has a weight parameter $\lambda_n$ that controls the

significance of the feature. The model can be expressed by the equation

$$\hat{e} = \underset{e}{argmax} \sum_{n=1}^{N} \lambda_n h_n(e,f) \tag{2.7}$$

We can see that the noisy-channel model is a special case of the log-linear model if we use the feature functions

$$h_1(e,f) = log\, P(e) \tag{2.8}$$

$$h_2(e,f) = log\, P(f|e) \tag{2.9}$$

and set $\lambda_1 = \lambda_2 = 1$. The probability $P(e)$ corresponds to the language model and the probability $P(f|e)$ corresponds to the translation model. However, we use logarithm of the probabilities instead.

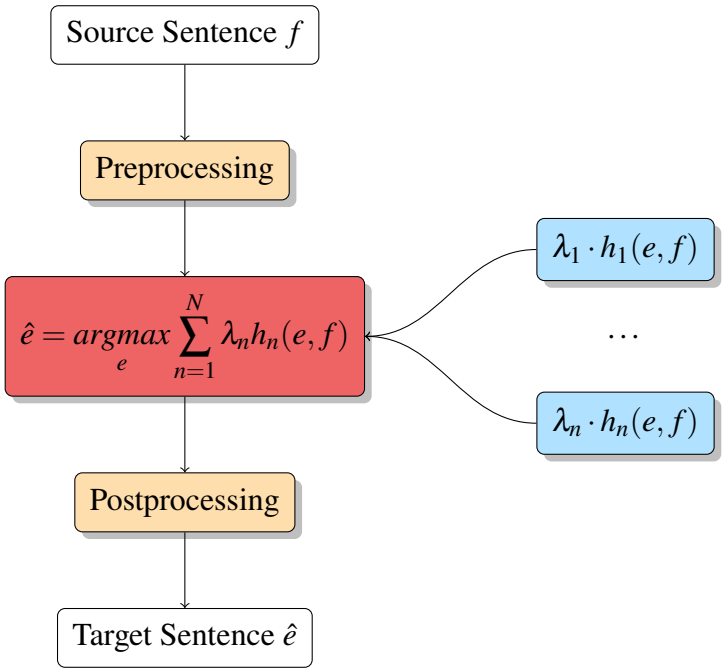



Figure 2.2: Processing pipeline of an MT system based on the log-linear model which can use various features functions $h_n(e,f)$ to find the best translation.

The log-linear model allows to define many different feature functions (see Figure 2.2) that take input sentence $f$ and output sentence $e$ as parameters. Just to mention some functions which are used in current statistical MT systems:

- length penalty function that penalizes short or long sentences,
- additional language model based on part-of-speech tags,

- inverse translation probability $P(e|f)$,
- distortion model that controls how far words can change position in the sentence,
- syntactic or shallow semantic features.

The feature weights $\lambda_n$ need to be optimized so that the model performs well. This optimization is not used in the noisy-channel model since the language and translation model have the same weight by default. Minimum error rate training (MERT) (Och; 2003) is used as a standard optimization method for tuning parameters $\lambda_n$. We discuss this topic in Section 2.2.6 more in detail.

### 2.2.3 Phrase-Based MT

One of the first statistical MT systems (Brown et al.; 1990) used word-based translation model, i.e. words in the source sentence were translated one by one. However, there exist sequences of words in almost every language that have a fixed translation. These sequences can be translated at once and increase the accuracy of the MT system. Phrase-based models take advantage of this observation.

Phrase-based models (Och and Ney; 2004) are based on extracting phrase translations, which are stored in *phrase tables*, from bilingual texts. This is done in several steps:

1. A word *alignment* between words in the source and the target language sentence is computed. A word alignment is a many-to-many relation between words that reflects which source word has the same meaning as a target word. It is possible that a source/target word is not aligned to any word in the other sentence. There can be also words that are aligned to several different words. For example English articles do not have a corresponding concept in Czech, therefore *the* and *a* are usually unaligned or they are attached to some other word. The word alignment can be extended to cover also unaligned words to get better translation performance (Och and Ney; 2003).

2. Individual phrase translations are extracted by extending the word alignment to phrase alignment. This is done by considering all contiguous sequences of words in the source and the target sentence so that words in the source/target phrase align only to words within the target/source phrase. Since we require that a phrase is a contiguous sequence of words, it cannot happen that a word in the middle of a phrase aligns to a word outside of the corresponding phrase in the other language. After the phrase extraction is finished, the phrase translation probabilities are estimated using the phrase co-occurrence counts.

Figure 2.3 shows an example of a sentence translated by a phrase-based MT system. Short phrases as *The economics profession* are translated as one block. Therefore, the noun-adjective agreement in case, number and gender between the Czech target words *ekonomická* and *profese* can be easily preserved.

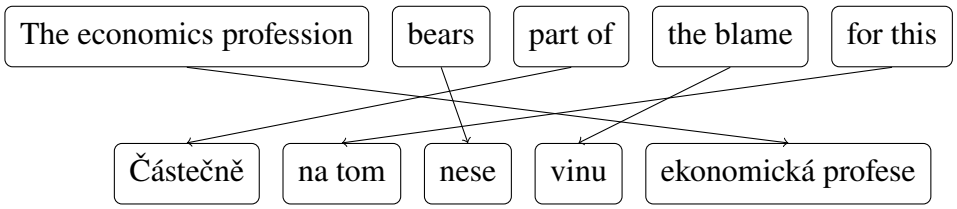



Figure 2.3: Phrase-based MT systems create translation by concatenating individual phrase translations (with possible reordering of phrases).

The biggest advantage of phrase-based models is that they can capture local dependencies between words by translating whole phrases instead of single words. The result is that short phrases are translated quite accurately. However, grammatical relations between distant words are difficult to handle with phrase-based models. If we consider the sentence


(2.10)

we can see a long distance relation between the noun *book* and the verb *returned*. This relation must be reflected by grammatical agreement, which is expressed by appropriate ending of the verb in Czech. However, the words are too distant to constitute one phrase in a phrase-based model.

### 2.2.4 Training

The training of a statistical MT system is one of the most computationally intensive parts in the construction of an MT system. It requires estimation of model parameters as accurately as possible from large amounts of data.

The major problem is that the training data represents only a subset of all possible sentences in a language. Even if we collect as much training data as possible, we will still miss a lot of grammatically correct sentences that will not appear in our training data. Therefore, it is not possible to get a sufficient amount of information about the actual real-world probabilities and we can only compute rough estimates. However, this is often enough to construct a statistical MT system that performs reasonably well.

The data used for MT training needs to be bilingual and sentence-aligned. This means that we need two texts, one in the source language and the other one in the target language, they must be segmented into sentences, and each sentence in one language must have a counterpart in the other language. Figure 2.4 shows an example of sentence-aligned training data.

| **English** | **Czech** |
|---|---|
| Such a strategy simply defies common sense . | Takové strategie prostě popírají zdravý rozum . |
| A Breakthrough Against Hunger | Průlom v boji proti hladovění |
| Historically , the stock market has performed well . | Historicky si akciový trh vedl dobře . |
| Education will be similarly transformed . | Podobně se transformuje školství . |
| Most likely , he would have done worse . | S největší pravděpodobností by se mu dařilo hůře . |
| That seems a wise decision . | Zdá se , že je to moudré rozhodnutí . |
| But the status quo is unacceptable . | Současný stav je ovšem nepřijatelný . |
| Other central banks make similar claims . | Jiné centrální banky vydaly podobná prohlášení . |

Figure 2.4: Example of sentence-aligned training data.

The reason why we require that the training texts are segmented into sentences is simple: it is much easier to find alignment of words within a sentence containing twenty words rather than in the whole training data which can consist of millions of words. We can make this simplification because a sentence can typically stand alone without substantial syntactical relationship to other sentences. Of course, we lose some information about the context in which a sentence was uttered, but this is a reasonable trade-off since it allows to dramatically simplify the MT model. Moreover, the simplification allows to translate sentences one by one without any relationship to other sentences. This can be used for parallelization of the translation process which leads to significant speedup. Another reason why to work on the sentence level is that computers can quite easily recognize the end of the sentence because it is explicitly marked in the text by full stop. Therefore, it is possible to automatically extract individual sentences from the text with relatively high accuracy.

Training of statistical MT models is based on estimating probabilities. Usually, the probabilities are computed using relative frequency counts. To illustrate this method better, we estimate the translation probabilities for the phrase *a cat* in Figure 2.5. We compute counts of individual translation options for the phrase using our training data. These counts are listed on the left side of Figure 2.5. To compute the translation probabilities we simply divide the count of the individual translation by the total number of translation options weighted by the number of their occurrences. This method is very intuitive and easy to use. It represents a fundamental method that is used in statistical MT system training.

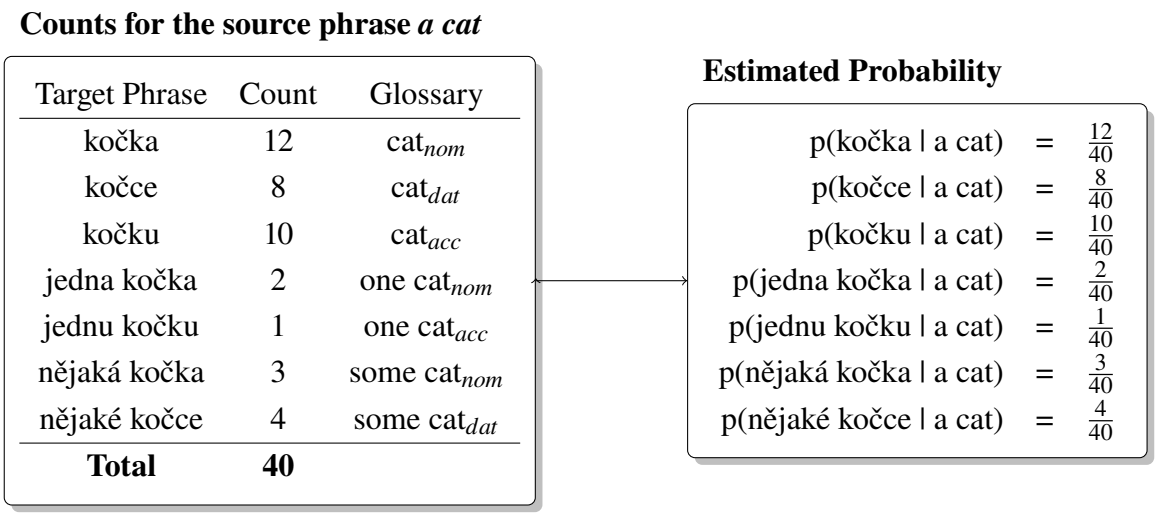



Figure 2.5: Translation probabilities are computed from training data using relative frequency counts.

### 2.2.5 Decoding

The process of translating a source language text is called *decoding* in statistical MT. It relates to the noisy-channel model described in Section 2.2.1, which models the extraction of the original sentence from a corrupted text, i.e. the "decoding" of the sentence. In this section, we describe the decoding procedure more in detail.

Decoding uses probabilities estimated during the training phase. Usually, each sentence is segmented into small chunks, i.e. sequences of words, that are translated one by one and then combined together. However, every small chunk can have multiple correct translations because it can be expressed in different ways in the target language. Moreover, the longer the sentence the more chunks need to be translated. If we wanted to find the best translation of a sentence, we would have to search all combinations.

While the training of an MT system can take several hours up to several days of intensive parameter optimization because it is done only once, the actual translation of a sentence is more time critical. We do not want to wait for the translation of one sentence for hours or even days. Therefore, approximation methods have to be used in order to compute the resulting translation fast enough. Instead of searching all possible combinations, we try to sort out combinations that are probably not a good translation. We can thus reduce the search space and can find the best translation faster.

Nevertheless, it is possible that we discard a partial translation which we think is not very good, but it is actually part of the best translation. Thus, we can lose the best translation on the way during the decoding. This problem can be partially solved by keeping not only the best partial

translation but several best translations. The partial translations are stored on the decoding stack, and they are expanded step by step.

After the whole sentence has been processed we can output not only the best translation but also a set of the best scoring translations. They constitute the so called *n-best lists*. The n-best lists typically contain tens or hundreds of translations. Some statistical MT systems output complete n-best lists and use an external scorer to select the best translation of the input sentence from the n-best list.

In recent years, experiments with combining output of several MT systems were conducted. The idea is that some system could be good at a specific translation subtask, e.g. maintaining the target sentence grammaticality or having better lexical choice, but the overall quality can be mediocre because of bad performance on some other subtask. Smart combination of several MT system outputs could therefore lead to better overall quality. However, the results (Callison-Burch et al.; 2009) are inconclusive about the improvement in translation quality.

### 2.2.6 Model Parameter Optimization

Optimization of model parameters is an important part of the MT system training pipeline. Each feature function of the log-linear model has its own weight that needs to be optimized in order to reflect its impact on the translation quality. The Minimum Error Rate Training (MERT) algorithm is an established algorithm to get the optimal weights for the log-linear model.

#### Minimum Error Rate Training

The minimum error rate training algorithm (Och; 2003) is a widely used method in statistical MT to optimize log-linear model parameters. This process, which is called *tuning*, is generally applied after the training of the individual feature functions and before the actual deployment of the translation system (see Figure 2.6). The goal of this optimization phase is to maximize the performance of the MT system on a small sample of data, called the *development set*, in order to improve the performance on unseen data as well.

Given model weights $\lambda_i$ ($i \in 1 \ldots N$), the algorithm performs a one-dimensional line minimization along the $i$-th dimension in an $N$-dimensional error space, defined by some MT quality metric. The line minimization along dimension $i$ represents optimizing the parameter $\lambda_i$, while keeping all other parameters fixed. This search is repeated along all dimensions until the local optimum is found. To avoid finding a poor local optimum, the algorithm starts from several random initial parameter values. In order to reflect the change in translation quality due to the altered model weights, the development set is repeatedly translated and the search for better parameters is continued until no improvement in translation quality can be reached.

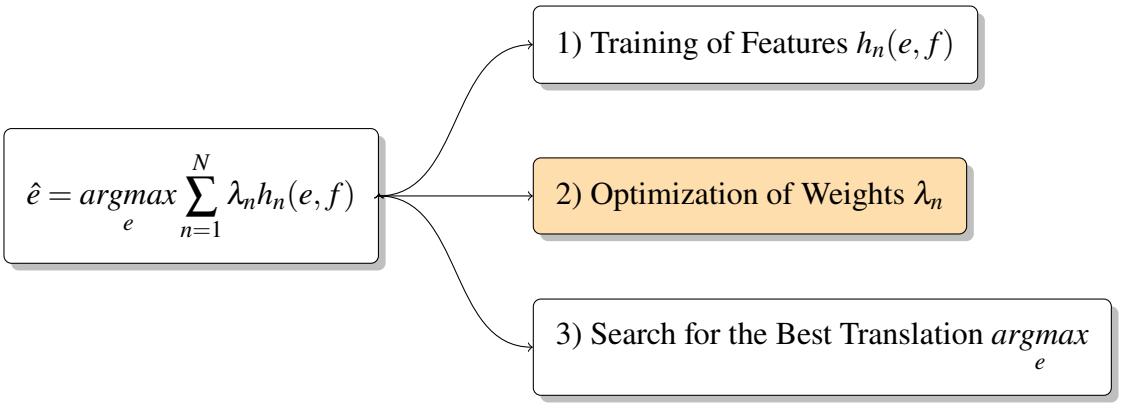



Figure 2.6: The statistical MT system training can be decomposed into 1) training of feature functions and 2) optimization of model parameters. Then, the system is ready for 3) decoding.

This method, however, requires a reliable automatic measure of translation quality for the optimum search. The metric should correlate to human assessment of MT quality. This is a difficult task because even humans sometimes do not agree which translation is better. Moreover, it should be easy and fast to compute so that it does not slow down the MERT algorithm.

### Metrics

Almost all MT quality metrics[2] follow the same principle: they compute some similarity score to a human reference translation. This approach proved to provide the best results when evaluating MT system output, even thought it requires a reference set of translations to be manually translated by human translators. Nevertheless, the translation can be done only once and it can be reused many times if we guarantee that the reference set is not included in the training data for MT systems. Generally, only one reference translation is sufficient to compute the metric score but some metrics take advantage of several human translations of the same sentence. This can help the metric to compute a more reliable score because one sentence can often be translated in many different ways with the same meaning.

One of the first MT metrics with an acceptable human judgment correlation was BLEU (Papineni et al.; 2002). It is based on the geometric mean of n-gram precision and uses brevity penalty to penalize candidate translations that are shorter than the reference. Since its introduction, BLEU has established as the golden standard in MT quality evaluation. Therefore, the first implementations of MERT used BLEU as the MT quality metric.

Many other MT quality metrics have been designed. The development of new metrics is

[2]We understand under MT metric only some measure of translation quality that can evaluate MT system output. There is no direct relation to metrics and metric spaces known from the field of mathematical analysis.

still ongoing and the metrics' performance is regularly measured in global evaluation campaigns (Przybocki et al.; 2008; Callison-Burch et al.; 2010). It is out of scope of this thesis to name all MT metrics, but we provide a short list of the best known ones:

- WER (Su and Wu; 1992),
- PER (Tillmann et al.; 1997),
- NIST (Doddington; 2002),
- GTM (Turian et al.; 2003),
- Meteor (Banerjee and Lavie; 2005),
- TER (Snover et al.; 2006),
- CDER (Leusch et al.; 2006) or
- SemPOS (Kos and Bojar; 2009).

Any of them can be in principle used in MERT instead of BLEU to find the optimal model parameters.

# Chapter 3

# Rich Annotation in MT Evaluation

In this chapter we investigate the effect of using an MT quality metric which takes advantage of rich annotation as the translation quality metric in the MERT algorithm.

## 3.1 Motivation

Using a metric that requires rich annotation of the text, e.g. part of speech or lemma, can help to evaluate the text on a more suitable level of detail for the given language. We can work on the level of lemmas instead of the surface forms if the language contains too many forms of one word, and the translation system is not able to generate the correct word forms. The task to translate to the correct lemma is easier for the translation system. The metric can give credit at least for correct lemmas if the system was not capable of generating the correct word forms.

On the other hand, it is possible to use rich annotation to obtain more fine-grained information about the translated words. We can analyze the translated sentences by available annotation tools and check that the sentences have appropriate grammatical structure or that all words are assigned the correct part of speech.

Thus, rich annotation gives us the opportunity to evaluate the translated text more accurately than metrics working only on surface forms. In this thesis we use a metric called SemPOS, which works on the level of lemmas to overcome the data sparseness problem for Czech as a rich morphology language, and which segments words into categories given their semantic part of speech to increase the evaluation accuracy.

## 3.2 SemPOS

The MT evaluation metric SemPOS (Kos; 2008; Kos and Bojar; 2009) is inspired by metrics based on overlapping of linguistic features in the reference and in the translation (Giménez and Márquez; 2007).[1] It operates on so-called "tectogrammatical" representation of the sentence (Sgall et al.; 1986; Hajič et al.; 2006), a deep-syntactic dependency tree that includes only content words as nodes. SemPOS disregards the syntactic structure and uses semantic part of speech (sempos) of the words (noun, verb, adjective, etc.). There are 19 fine-grained part-of-speech classes. For each semantic part of speech the overlapping is computed as

$$\text{overlapping}(t) = \frac{\sum_{i \in I} \sum_{w \in r_i \cap c_i} \min(\text{count}(w,t,r_i), \text{count}(w,t,c_i))}{\sum_{i \in I} \sum_{w \in r_i \cup c_i} \max(\text{count}(w,t,r_i), \text{count}(w,t,c_i))} \tag{3.1}$$

The semantic part of speech is denoted $t$; $c_i$ and $r_i$ are the candidate and reference translations of sentence $i$, and $\text{count}(w,t,rc)$ is the number of words $w$ with type $t$ in $rc$ (the reference or the candidate). The matching is performed on the level of lemmas, i.e. no morphological information is preserved in $w$s.

The final SemPOS score is obtained by macro-averaging over all parts of speech:

$$\text{SemPOS} = \frac{1}{|T|} \sum_{t \in T} \text{overlapping}(t) \tag{3.2}$$

where $T$ is the set of semantic part-of-speech types.

### 3.2.1 Correlation with Human Judgments

Kos and Bojar (2009) and Bojar et al. (2010) report that SemPOS performs generally better than BLEU, NIST or GTM (all of them are metrics working on surface forms) on Czech. The reason for this is illustrated in Figure 3.1 and Figure 3.2. SemPOS ignores word order, and all auxiliary words (prepositions, particles of complex verbs) are omitted. Only base forms (lemmas) of content words are considered. The risk of not giving credit for a word due to a difference in morphological form and of unjustified overestimation due to random agreement in sequences of tokens is thus much lower.

Bojar et al. (2010) also evaluate the performance of the linear combination of SemPOS and BLEU. The BLEU score should compensate for one of the major drawbacks of SemPOS: SemPOS completely ignores the word order, which is too coarse even for languages with relatively free word

[1] There are two versions of this article which differ in the overlapping formula. We use the formula from the article which is available at one of the author's homepage `http://www.lsi.upc.edu/~jgimenez/pubs.html`.

order like Czech. Thus, weighted linear combination of SemPOS and BLEU (computed on lemmas) should compensate for this. The reported correlation for the linear combination of SemPOS and BLEU was indeed slightly better on Czech that for the plain SemPOS score.

| | |
|---|---|
| SRC | Congress yields: US government can pump 700 billion dollars into banks |
| REF | kongres ustoupil : vláda usa může do bank napumpovat 700 miliard dolarů |
| sys1 | kongres \| výnosy \| : vláda usa může \| čerpadlo \| 700 miliard dolarů \| v \| bankách |
| sys2 | kongres vynáší : us vláda může čerpat 700 miliardu dolarů do bank |

Figure 3.1: Too much focus on sequences in BLEU: *sys2*'s output is better but does not score well. BLEU gave credit to *sys1* for 1, 3, 5 and 8 fourgrams, trigrams, bigrams and unigrams, resp., but only for 0, 0, 1 and 8 *n*-grams produced by *sys2*. Confirmed sequences of tokens are underlined and important errors (not considered by BLEU) are framed.

| | |
|---|---|
| REF | kongres/n ustoupit/v :/n vláda/n usa/n banka/n napumpovat/v 700/n miliarda/n dolar/n |
| sys1 | kongres/n výnos/n :/n vláda/n usa/n moci/v čerpadlo/n 700/n miliarda/n dolar/n banka/n |
| sys2 | kongres/n vynášet/v :/n us/n vláda/n čerpat/v 700/n miliarda/n dolar/n banka/n |

Figure 3.2: SemPOS evaluates the overlap of lemmas of content words given their semantic part of speech (n, v, ...). Underlined words are confirmed by the reference.

## 3.3 Experiments

In our experiments we use SemPOS and the linear combination of SemPOS and BLEU as translation quality metrics in MERT to tune for better model parameters. By using SemPOS as the MERT optimization metric we would like to find better model weights that would better reflect human assessment of MT quality and provide MT system output of higher quality.

### 3.3.1 System Configuration

We trained the phrase-base statistical MT system Moses[2] on two datasets taken from the CzEng 0.9 corpus (Bojar and Žabokrtský; 2009):

[2]Moses is an open-source phrase-based decoder that implements the beam-search algorithm to find the best translation of a source sentence. It is widely used by the MT research community for developing and testing new features.

- **Small** - consisting of the news domain of CzEng,
- **Large** - consisting of all CzEng domains except `navajo` and additionally the EMEA corpus (Tiedemann; 2009).

Figure 3.3 provides word and sentence counts for both datasets. We use WMT10[3] development sets for model parameter optimization (news-test2008) and evaluation (news-test2009). The BLEU scores reported in this thesis are based on truecased word forms in the original tokenization as provided by the decoder. The ± value given with each BLEU score is the average of the distances to the lower and upper empirical 95% confidence bounds estimated using bootstrapping (Koehn; 2004).

| | Sentences | Words | |
|---|---|---|---|
| | | English | Czech |
| Small | 126,144 | 2,883,893 | 2,645,665 |
| Large | 7,544,465 | 89,135,590 | 79,192,822 |

Figure 3.3: Statistics for the Small and Large training datasets.

We use the following Moses configuration:

- Standard GIZA++ word alignment based on both source and target lemmas.
- Two alternative decoding paths (forms always truecased):
  - form+tag → form,
  - form → form.

  The first path is more specific and helps to preserve core syntactic elements in the sentence. The second path serves as a back-off.
- One 5-gram Czech language model of truecased forms trained on the Czech part of the datasets using SRILM (Stolcke; 2002).
- Lexicalized reordering (`orientation-bidirectional-fe`) based on forms.

Additionally, we applied significance filtering (Johnson et al.; 2007) of phrase tables (only) for the Large dataset, because the phrase tables were too large to fit into memory. We set filter value to `a+e` and the cut-off threshold to `20`.

[3] `http://statmt.org/wmt10/`

In our experiments we use Z-MERT (Zaidan; 2009), a recent implementation of the MERT algorithm, instead of the standard Moses MERT. We implemented a wrapper script *zmert-moses.pl* to launch Z-MERT. It supports most of the parameters of the older script *mert-moses-new.pl*. More details about the supported parameters are included in Appendix B.

### 3.3.2 Integration into MT Pipeline

The SemPOS metric requires to assign the (deep-syntactic) lemma and sempos tag to all autosemantic words. The original SemPOS implementation uses an external annotation tool TectoMT (Žabokrtský and Bojar; 2008)[4] for the linguistic processing. TectoMT follows the complete pipeline of tagging, surface-syntactic analysis and deep-syntactic analysis, which is the best but rather costly way to obtain the required information. Since we need to analyze whole n-best lists in the MERT algorithm, this process would be too time consuming.

Instead, we apply TectoMT to the *training data*, express the (deep) lemma and sempos as additional factors using a blank value for auxiliary words, and use Moses factored translation to translate from English forms to triplets of Czech form, deep lemma and sempos. Factored translation models (Koehn et al.; 2007) allow words to be vectors of features.

Figure 3.4 shows that this approach can save significant amount of time compared to the TectoMT analysis, where one MERT experiment can last more than three days without any improvement in the final score. The SemPOS scores for the TectoMT path are lower and the BLEU scores more unstable than for the factored translation, which suggests that the TectoMT analysis penalizes ungrammatical sentences whereas factored translation blindly assigns the most probable lemma and sempos tag to each word.

| | BLEU | SemPOS | Runtime | Iters |
|---|---|---|---|---|
| TectoMT | 9.80±0.40 | 29.11 | 3d15h59m | 20 |
| | 7.88±0.37 | 29.16 | 3d09h30m | 20 |
| Factored | 9.20±0.45 | 29.41 | 11h29m | 20 |
| translation | 10.03±0.41 | 30.45 | 8h37m | 20 |

Figure 3.4: Processing of n-best lists with TectoMT is very time consuming compared to factored translation. The table shows two MERT runs for each path optimized towards SemPOS, with attributes for SemPOS provided either after the translation (TectoMT) or generated during the translation (factored translation). Translation and annotation (only in TectoMT) tasks were run on 15 CPUs in parallel.

[4] http://ufal.mff.cuni.cz/tectomt/

## 3.4 Results

The obtained results suggest that optimizing model parameters towards SemPOS does not bring the desired improvement in translation quality. We use the SemPOS and BLEU metrics to measure the quality of the final system output.

### Small Data

Figure 3.5 shows BLEU and SemPOS scores for the Small data. The first two lines in the table show the metric scores for the system optimized towards BLEU, and the following two lines show metrics scores for the system optimized towards SemPOS.

| Weights | | Scores | |
|---|---|---|---|
| BLEU | SemPOS | BLEU | SemPOS |
| 1 | 0 | 10.44±0.41 | 30.89 |
| 1 | 0 | 10.41±0.41 | 30.74 |
| 0 | 1 | 10.03±0.41 | 30.45 |
| 0 | 1 | 9.20±0.45 | 29.41 |
| 1 | 1 | 9.58±0.43 | 30.36 |
| 2 | 1 | 10.17±0.39 | 30.48 |
| 3 | 1 | 10.27±0.44 | 30.96 |
| 5 | 1 | 10.29±0.42 | 30.58 |
| 1 | 2 | 9.82±0.41 | 30.56 |
| 1 | 3 | 9.28±0.40 | 30.18 |
| 1 | 5 | 8.92±0.40 | 29.75 |

Figure 3.5: Optimization towards BLEU, SemPOS and the linear combination of SemPOS and BLEU on the Small dataset. The best results are obtained when optimizing towards plain BLEU, or a linear combination of SemPOS and BLEU with the dominance of BLEU.

It is interesting that both metric scores drop if we use SemPOS for tuning the model parameters. This could be explained that SemPOS is not stable in evaluating the MT output and misleads the MERT algorithm into choosing suboptimal parameter values.

The best BLEU score (10.44) is obtained when we optimize towards plain BLEU. However, the best SemPOS score (30.96) is not obtained when optimizing towards SemPOS but towards a linear combination of BLEU and SemPOS with weights 3:1. The explanation could be that BLEU maintains the stability and SemPOS helps a little bit with lexical choice of words. Nevertheless,

the improvement in SemPOS score is only marginal compared to one of the scores obtained for plain optimization towards BLEU (30.89).

All linear combinations with the domination of SemPOS over BLEU result in drop in both scores: the higher the weight of SemPOS the higher the drop. This supports the hypothesis that SemPOS destabilizes MERT and prevents it from finding the optimal values even for SemPOS itself.

**Large Data**

In general, the instability of SemPOS is not as high on the Large data as on the Small data. Figure 3.6 shows that the change in SemPOS scores is rather moderate. Moreover, the highest SemPOS scores are achieved when optimizing towards SemPOS or a linear combination of SemPOS and BLEU with the dominance of SemPOS.

| Weights | | Scores | |
|---|---|---|---|
| BLEU | SemPOS | BLEU | SemPOS |
| 1 | 0 | 12.77±0.47 | 32.19 |
| 1 | 0 | 12.62±0.46 | 31.75 |
| 0 | 1 | 11.06±0.40 | 32.80 |
| 0 | 1 | 11.37±0.41 | 32.24 |
| 1 | 1 | 12.42±0.46 | 32.63 |
| 2 | 1 | 12.71±0.47 | 32.34 |
| 3 | 1 | 12.75±0.47 | 31.96 |
| 5 | 1 | 12.84±0.50 | 32.18 |
| 1 | 2 | 12.49±0.47 | 32.67 |
| 1 | 3 | 12.08±0.43 | 32.77 |
| 1 | 5 | 12.34±0.45 | 32.39 |

Figure 3.6: Optimization towards BLEU, SemPOS and the linear combination of SemPOS and BLEU on the Large dataset. The best BLEU score (12.84) is obtained for the linear combination of SemPOS and BLEU with weights 1:5, however, the score is not significantly better than when optimizing towards plain BLEU (12.77). The best SemPOS score (32.80) is obtained when using plain SemPOS as optimization metric.

However, when we optimize towards plain SemPOS we can still see a significant drop in BLEU score: from 12.77 to 11.06. On the other hand, SemPOS scores stay relatively high even if we use plain BLEU as the optimization metric. This leads to the observation that optimization towards

plain BLEU leads to a relatively high SemPOS score, but optimization towards plain SemPOS does not necessarily mean a high BLEU score.

The linear combination of SemPOS and BLEU in MERT does not bring any significant change in metric scores. Nevertheless, we can observe that the inclusion of BLEU in the linear combination stabilizes the overall scores of both metrics.

# Chapter 4

# Source-Context Model

In this chapter we describe an extension of the log-linear model with the source-context model. This extension should help the decoder to prefer more suitable translations in the context of the source sentence and improve the translation quality.

The source-context model consists of a set of individual source-context features. We understand under a source-context feature any function that takes a source sentence and a translation phrase pair (consisting of a source and a target phrase) as input and returns some score for the phrase pair in the context of the source sentence. We assume that the source phrase is a substring of the sentence.

We restrict the context only to the current source sentence. It would be possible to consider wider context as surrounding source sentences or even the whole source document. However, this approach would not be compatible with the decoder we use, which considers sentences as separate units and translates them individually. Target side context would be also a potential way to improve translation quality but we could evaluate only the context of the target text which has been generated so far.

## 4.1 Motivation

The biggest advantage of the source-context model is that it can make more fine-grained decisions than the basic log-linear model based only on the translation model. Without the source-context model the decoder evaluates each phrase translation only in global context, i.e. it cannot differentiate whether a phrase has a specific translation in a given context, and it must rely on the language model that it will be able to sort out the improbable translations.

We see the following potential benefits of the source-context model:

- **Better lexical choice**

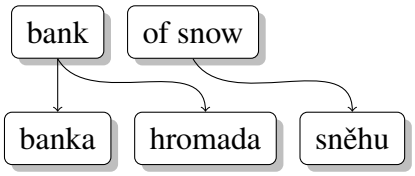
  The source sentence can provide valuable information about the context in which the source phrase was uttered. Therefore it is possible to select a better translation. We can take the word *bank* and its possible translations (see Section 1.1) as an example.

  Let's assume that we need to translate the phrase *bank of snow* to Czech, which would be translated as *hromada sněhu*, but we have not seen the phrase *bank of snow* in our training data. Therefore we need to constitute the phrase from smaller phrases consisting of individual words. If we always took the most probable translation of each word, we would most certainly get a translation *banka sněhu*, since *bank* usually means a financial institution. However, this is not the desired translation.

  With the source-context model we can take the surrounding words, e.g. *snow*, and look whether they occur near the word *bank* in the corpus. We will get several occurrences of *snow bank* with appropriate translation *bank* → *hromada*, but no translation *bank* → *banka*. This can force the decoder to select the preferred translation option *hromada* instead of the globally preferred option *banka*, if the source-context feature overweighted the translation model. See Figure 4.1 for illustration.

a)

| | | | |
|---|---|---|---|
| bank | → | banka | 0.9 |
| bank | → | hromada | 0.1 |
| | ... | | |
| of snow | → | sněhu | 1.0 |




b)

| Source-context corpus | |
|---|---|
| English: | ... snow bank ... |
| Czech: | ... hromada sněhu ... |

c)

| *bank* | TM | SCX | TM × SCX |
|---|---|---|---|
| banka | 0.9 | 0.1 | **0.09** |
| hromada | 0.1 | 1.0 | **0.1** |

Figure 4.1: Illustration of the influence of a source-context feature which uses surrounding words to promote more suitable translations; a) phrase table, b) example of a co-occurrence of *bank* and *snow* in the source-context corpus, c) feature values for the translation model (TM) and the source-context feature (SCX) with the resulting scores, which prefer *hromada* as a translation of *bank*. The feature score SCX for *banka* is low because this word does not occur in the source-context corpus near *snow*.

- **Better selection of word forms**
  Selected source-context features can help the MT system to generate better word forms of translated words. They can filter out words forms in the target language that are not suitable in the given source sentence context. We show this effect on the sentence *you see a cat* which we would like to translate to Czech. In English, there is fixed word order: subject, in nominative, generally precedes the content bearing verb, and object, in dative or accusative, follows the verb in a normal sentence. We can use this observation when we translate the phrase *a cat*. Because it follows a content verb, the Czech translation will be probably in some other case than nominative and the source-context feature can reflect this fact. This situation is depicted in Figure 4.2: the translation table probabilities prefer *kočka* in nominative because it appeared most of the time in nominative in the training corpus. On the other hand the source-context feature uses the knowledge about the preceding word to alter the probabilities in favor of $kočce_{dat}$ and $kočku_{acc}$. Thus, the preferred translation $kočku_{acc}$ has a better chance to be chosen by the decoder.

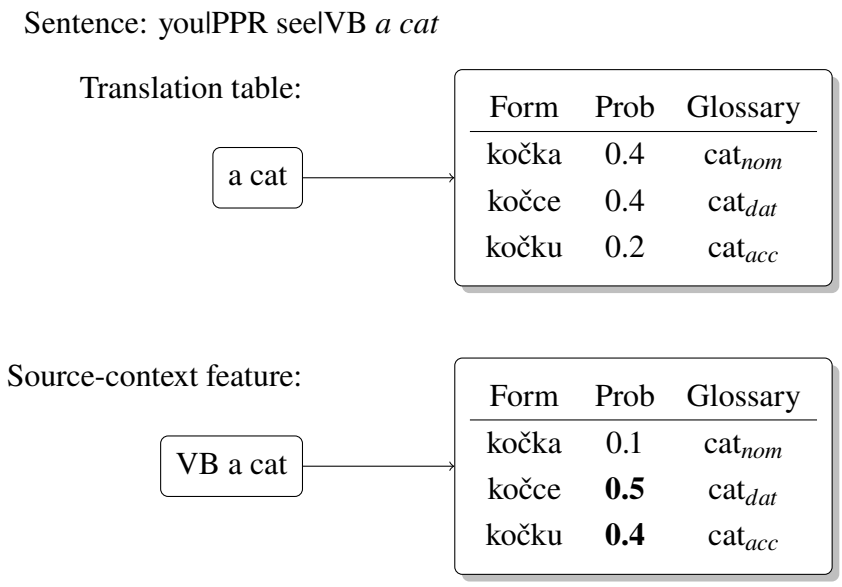



Figure 4.2: Source context can support better word form choice on the target side.

We showed two motivation examples that the source-context model in form of source-context features can improve phrase-based MT. Before we start with the description of the source-context model more in detail we provide an overview about past approaches to source context in MT in Section 4.2. Then, we describe the implemented features in Section 4.3, and provide some implementation details in Section 4.4. Finally, we discuss our experiments in Section 4.5 and provide results in Section 4.6.

## 4.2 Past Work

Several approaches to source context in MT have been examined in recent years. They range from incorporating a word sense disambiguation (WSD) module into statistical MT decoder to help with better lexical choice. Other approach, inspired by the WSD techniques, decomposed the external WSD module into many separate source-context features and used them directly in the log-linear model. Finally, source context has been also used to rerank n-best lists to select the best sentence translation.

### 4.2.1 Word Sense Disambiguation

Even though WSD represents a separate research field in natural language processing, it can help machine translation to some extent, because it tries to solve a similar subproblem as MT does: we want to find the most accurate meaning/translation of a word/phrase in a given context both in WSD and MT. Therefore, it is reasonable to use a technique which performs well in WSD to improve MT quality. Currently, state-of the-art WSD systems can provide quite accurate results, which makes them a promising approach to improve MT.

However, there are several differences between WSD and MT. WSD tries to get the most plausible meaning only of one word, while we want to get the best translation for a sequence of words in MT. Moreover, WSD is typically a classification task which has a predefined sense inventory. On the other hand, MT systems need to be capable of translating any arbitrary sentence and common dictionaries and phrase tables do not often contain all possible translations of a word. Therefore the MT task is much more difficult than plain word sense disambiguation.

#### WSD for Phrase-Based MT

One of the first experiments with WSD in statistical MT was done by Carpuat and Wu (2005). They incorporated a dedicated WSD module into a phrase-based MT system. They used two different approaches:

- First, they used the WSD module to restrict possible translation options for a given phrase to agree with the WSD prediction of some source word. Thus, they restricted the set of translation options filtering out phrases that contained translations inconsistent with the WSD module.
- Second, they used WSD in post-processing of a sentence when they directly replaced a translated word by the WSD prediction.

The results for Chinese to English translation were not very promising because none of the approaches did help to increase the MT quality. Since the WSD module was limited only to individual words, it did not have any significantly positive effect on the translation quality. Generally, the WSD module decreased the performance of the MT system with the exception of a few words that were disambiguated better. The authors modified their approach from disambiguation of single words to disambiguation of whole phrases in Carpuat and Wu (2007) and reported improvement in translation quality for the same language pair (see Section 4.2.2).

### WSD for Hierarchical MT

Another branch of research was focused on hierarchical MT systems. Chan et al. (2007) integrated an external WSD system into a hierarchical MT system, Hiero Chiang (2007). Hiero implements the synchronous context-free grammar (CFG) formalism. Synchronous CFG is an extension of the classical context-free grammar to pairs of languages. The elementary structures in synchronous CFG are rewrite rules with aligned pairs of right-hand sides

$$R \rightarrow \langle \gamma, \alpha, \sim \rangle \tag{4.1}$$

where R is a nonterminal, $\gamma$ and $\alpha$ are strings of terminals and nonterminals, and $\sim$ is a one-to-one correspondence between nonterminals in $\gamma$ and $\alpha$. A synchronous CFG derivation begins with a pair of linked start symbols. At each step, two linked nonterminals are rewritten by the components $\gamma$ and $\alpha$ from the right-hand side of a rule until no rule can be applied. Hiero follows the log-linear model with feature functions defined on derivations covering the source sentence.

Chan et al. (2007) defined probability

$$P_{wsd}(t|s) \tag{4.2}$$

of the WSD classifier and added this probability as a new feature to the log-linear model. $P_{wsd}(t|s)$ is the contextual probability of choosing $t$ as a translation for $s$ where $t$ is some substring of terminal symbols in $\alpha$ from rule $R$, and $s$ some substring of terminal symbols in $\gamma$. This probability was defined only on *nouns*, *verbs* and *adjectives*, which were part of $s$. Therefore the WSD module did not cover all possible synchronous CFG rules, and penalized[1] rules for which the probability was defined. Thus, it was necessary to add a second feature that would reward rules suggested by the WSD classifier

$$Pty_{wsd} = exp(-|t|) \tag{4.3}$$

[1]Remember that the log-linear model tries to maximize sum of probability logarithms. Since the highest value for probability is one, the logarithm is never a positive value.

where $|t|$ is the length of the suggested translation $t$, to compensate for the negative value of the first feature.

The contextual probability $P_{wsd}(t|s)$ was computed for every rule $R$ considered during decoding using precomputed knowledge about lexical collocation, part-of-speech collocation, and wide context.

- **Lexical Collocations**
  Lexical collocation considered only local context. Three features were defined on surrounding tokens:
  - token immediately to the left of $s$,
  - token immediately to the right of $s$,
  - tokens immediately to the left and right of $s$.
- **Part-of-Speech Collocations**
  For parts of speech, features were defined on part of speech of tokens on the following positions relative to $s$:
  - token immediately to the left of $s$,
  - current token $s$,
  - token immediately to the right of $s$.
- **Wide Context**
  Wide context considered all unigrams (single words) in the surrounding context of $s$. These unigrams could be in a different sentence than $s$, however the authors do not provide details about the maximum span of this context. They performed feature selection on surrounding words by including a unigram only if it occurred three or more times in some sense of $s$ in the training data.

The authors reported statistically significant improvements in BLEU score over the baseline Hiero system for Chinese to English translation direction. Nevertheless, the achieved scores were only slightly better than the baseline system.

### 4.2.2 Disambiguation of Phrases

Later research suggests that if we disambiguate whole phrases instead of individual words we can obtain even better performance of MT systems.

### Dedicated WSD Module

Instead of forcing the decoder to match translations of individual words selected by a WSD module, Carpuat and Wu (2007, 2008) enhance a phrase-based MT system by a WSD model to provide a context-dependent probability distribution over the possible translation options for a given source phrasal lexicon entry.

The advantage of this approach is that the basic unit to disambiguate are whole phrases, instead of disambiguating individual words as in WSD. The sense candidates are provided by the baseline phrase tables. The probability distribution over the possible target translation candidates is computed using feature set containing:

- bag of all words in the source sentence,
- local collocations,
- position-sensitive local POS tags,
- basic dependency features.

Unfortunately, the authors are not specific enough how they used these features. They report that the contextual phrase-based lexicon improved the translation quality on a Chinese to English translation task. The results were consistently better for eight different MT metrics.

### Rich Log-Linear Features

One of the most straightforward ways to use source-context information during decoding is to define a set of features

$$p(e|f, f_{context}) \tag{4.4}$$

where $e$ is the target language phrase, $f$ is the source language phrase, and $f_{context}$ is the context of the source phrase in the sentence in which it was observed. Such features can be directly used in the log-linear model, and the model can automatically take care of finding correct weights for the features. This approach was chosen by Gimpel and Smith (2008) who directly added a set of feature functions to the log-linear phrase-based model. We will use the notation $f_k^l$ for the source phrase $f$ if we want to stress that is begins at position $k$ and ends at position $l$ in the sentence.

The feature functions were divided into following subsets:

- **Lexical Collocations**
  For a source phrase $f_k^l$, Gimpel and Smith (2008) included context of a the $m$-length sequence before it ($f_{k-m}^{k-1}$) and the $m$-length sequence after it ($f_{l+1}^{l+m}$). They used context lengths

for $m = \{1,2\}$, padding sentences with $m$ special symbols at the beginning and at the end. For each value of $m$, they included three features:

  - $p(e|f, f_{k-m}^{k-1})$, the left lexical context;
  - $p(e|f, f_{l+1}^{l+m})$, the right lexical context;
  - $p(e|f, f_{k-m}^{k-1}, f_{l+1}^{l+m})$, both sides.

- **Part-of-Speech Collocations**
  They used the same set of the lexical context features described above, but with POS tags replacing words in the context. They also include a feature which conditions on the POS tag sequence of the actual phrase being translated.

- **Syntax**
  The following parse tree features were used:

  - Is the phrase (exactly) a constituent?
  - What is the nonterminal label of the lowest node in the parse tree that covers the phrase?
  - What is the nonterminal label or POS of the highest nonterminal node that ends immediately before the phrase? Begins immediately after the phrase?
  - Is the phrase strictly to the left of the root word, does it contain the root word, or is it strictly to the right of the root word?

- **Position in Sentence**
  These features used information about the position of the phrase $f_k^l$ in the source sentence $f$, the phrase length, and the sentence length $n$.

  - Is the phrase at the start of the sentence ($k = 1$)?
  - Is the phrase at the end of the sentence ($l = n$)?
  - A quantization of $r = \frac{k+\frac{l-k+1}{2}}{n}$, the relative position in (0, 1) of the phrase's midpoint within $f$. They chose the smallest $q \in 0.2, 0.4, 0.6, 0.8, 1$ such that $q > r$.
  - A quantization of $c = \frac{l-k+1}{n}$ , the fraction of the words in $f$ that are covered by the phrase. They chose the smallest $q \in \frac{1}{40}, \frac{1}{20}, \frac{1}{10}, \frac{1}{5}, \frac{1}{3}, 1$ such that $q > c$.

The additional data which were required to compute the context features were extracted along with the phrase pairs during the execution of the standard phrase extraction algorithm, and the feature values were directly used during scoring the phrases by the decoder.

The selection of features was quite similar to features used by Carpuat and Wu (2007), although Gimpel and Smith (2008) also additionally included positional features.

Gimpel and Smith (2008) reported variable performance of individual feature types ranging across test sets and language pairs. For Chinese to English translation, statistically significant improvement was measured for lexical collocations on one test set, and POS collocations and syntactic features on another test set. The performance for German to English translation was mixed, sometimes causing decrease in translation quality.

### CCG Tags

Another approach to use source-context information in MT is mentioned by Birch and Osborne (2007) who used combinatorial categorial grammar (CCG) tags to include syntactical information into phrase-based MT system. They enriched the source side by CCG tags and used the tags as additional factor in factored translation.

Combinatorial categorial grammar is a formalism that assigns tags to individual words which reflect syntactical relations to other words. To put it simple, these tags contain information about additional words to fill required slots of a word to make it a well-formed sentence. For example a noun can be assigned a tag that says that a verb is required, since the simplest sentence consists of a noun and a verb. Words can be usually used in various syntactic structures and the same word can be assigned different CCG tags. For each word, we can then create a lexical entry with possible CCG tags.

There are two approaches to CCG tags in MT described in (Birch and Osborne; 2007). The first one suggests to use CCG tags on the target side and run an additional language model over them. However, we are interested only in the source side features in this thesis, hence we will discuss only the second approach. The authors extended an existing log-linear phrase-based MT system by adding the translation probabilities

$$P(t_w|s_{w+ccg}) \tag{4.5}$$

which represent the conditional probability of translating source phrase $s_{w+cgg}$ annotated with CCG tags to the target phrase $t_w$. They did not use back-off techniques to deal with sparse data but used a log-linear combination of feature functions defined over the source and target phrase.

The results obtained from experiments performed on German to English translation show that the feature functions defined over each source and target phrase should not be directly combined with the phrase-based log-linear model. They should be rather computed separately and only one cumulative score should be passed to the general log-linear model in order to improve the translation quality. The explication is that by including more but less informative features in one model, too much explanatory power may be transfered to the more specific features. In other

words, MERT training tends to overestimate the (negative or positive) weight for features that fire very rarely. Smith et al. (2005) demonstrated that using ensembles of separately trained models and combining them in a logarithmic opinion pool leads to better parameter values.

#### Trigger-Based Lexicon Model

Mauser et al. (2009) proposed the trigger-based lexicon model

$$p(e|f_1, f_2) \tag{4.6}$$

based on triplets of words consisting of two source words $f_1$, $f_2$ and one target word $e$ . The two source words can be viewed as triggers for a target word from the triplet. The triggers can originate from words of the whole source sentence, also crossing phrase boundaries of the conventional bilingual phrase pairs. The proposed model is able to capture long-distance effects such as verb splits or adjustments to lexical choice of the target word given the topic-triggers of the source sentence. The authors applied this model directly when scoring bilingual phrase pairs. Given a trained model for $p(e|f_1, f_2)$, they computed the feature score $h_{trip}$ of a phrase pair $(\bar{e}, \bar{f})$ as

$$h_{trip}(\bar{e}, \bar{f}, f_0^J) = -\sum_i log\left(\frac{2}{J \cdot (J+1)} \sum_j \sum_{j'>j} p(\bar{e}_i|f_j, f_{j'})\right) \tag{4.7}$$

where i moves over all target words in the phrase $\bar{e}$, the second sum selects all source sentence words $f_0^J$ including the empty word, and $j' > j$ covers the rest of the source sentence right of the first trigger. Negative log-probabilities were taken and normalized to obtain the final score (representing costs) for the given phrase pair. They trained the model on a subset of the overall training data because of the enormous number of triplets. The subcorpus contained 1.4M sentence pairs with 32.3M running words on the English side.

The authors reported a consistent improvement in BLEU and TER score for Chinese to English and Arabic to English translation over two test sets. This suggests that the context of the whole source sentence can disambiguate the target phrase without taking account of which part of the source sentence contributed to the target phrase.

### 4.2.3 Sentence-Level Disambiguation

#### Discriminative Lexicon Model

Mauser et al. (2009) also proposed a discriminative word lexicon model and integrated it into the standard phrase-based MT. The core of their model was a classifier that predicts target words, given the words of the source sentence. The structure of the source as well as the target sentence

was neglected in the model. They modeled the probability of the set of target words in a sentence $\mathbf{e}$ given the set of source words $\mathbf{f}$ . For each word in the target vocabulary, they calculated a probability for being or not being included in the set. The probability of the whole set was the product over the entire target vocabulary $\mathbf{V_E}$:

$$P(\mathbf{e}|\mathbf{f}) = \prod_{e \in \mathbf{e}} P(e^+|\mathbf{f}) \cdot \prod_{e \in \mathbf{V_E} \setminus \mathbf{e}} P(e^-|\mathbf{f}) \tag{4.8}$$

The event $e^+$ denotes that the target word $e$ is included in the target sentence and $e^-$ if not. They modeled the individual factors $P(e|\mathbf{f})$ of the probability in Equation 4.8 as a log-linear model using the source words from $\mathbf{f}$ as binary features

$$\phi(f,\mathbf{f}) = \begin{cases} 1 & \text{if } f \in \mathbf{f} \\ 0 & \text{else} \end{cases} \tag{4.9}$$

and lambda weights $\lambda_{f,.}$:

$$P(e^+|\mathbf{f}) = \frac{exp\left(\sum_{f \in \mathbf{f}} \lambda_{f,e} \cdot \phi(f,\mathbf{f})\right)}{\sum_{e^+,e^-} exp\left(\sum_{f \in \mathbf{f}} \lambda_{f,e} \cdot \phi(f,\mathbf{f})\right)} \tag{4.10}$$

The authors used the probability $P(\mathbf{e}|\mathbf{f})$ as an additional feature in the log-linear model. They argue that the discriminative word lexicon model, which completely disregards the structure in source and target sentences, is a suitable complement of the phrase model in phrase-based MT because it is able to predict global aspects of the sentence like tense or vocabulary changes in questions, while the phrase model is good in predicting translations in a local context. The results show similar performance, i.e. consistent improvement over the baseline system, as for the trigger-based lexicon model described in the section above.

#### *N*-best List Reranking

A completely different approach to source-context information was reported by Sudoh et al. (2008). Instead of using source-side information directly in the decoding process, they used a hierarchical system to generate an n-best list which was then reranked using source-context features. They used a large number of sparse binary features for the reranking, among them also context-dependent word pair features. These features were bag-of-words of both source- and target-side in the *previous* sentence, where target-side context words were extracted from n-best translation candidates.

The approach of Sudoh et al. (2008) is interesting because they used context words from the directly preceding sentence and not from the current sentence. Their motivation was to extract contextual information from dialogue utterances, when the word selection depends on the

previous utterances. However, the context features turned out to capture many general word co-occurrences and the reranker failed to distinguish better translation candidates from others.

## 4.3 Log-Linear Features

In this section, we discuss our approach to source context in phrase-based MT. We use a similar approach as Gimpel and Smith (2008) who defined a wide range of log-linear feature functions and integrated them directly into the log-linear model.

### 4.3.1 Collocation Features

Gimpel and Smith (2008) reported that the most useful source-context feature used part-of-speech tags of the surrounding words of a phrase. It requires that the source sentence is analyzed by a tagger and all words are assigned a POS tag.

#### Rich Factors

We decided to extend this approach to additional factors that could also bring an improvement in MT quality. We used the TectoMT platform to annotate the Small and Large datasets[2] mentioned in the previous chapter with the following factors:

- **pos** - part of speech,
- **lemma** - tectogrammatical (deep) lemma,
- **formeme** - contains information about the morphosyntactic form of a word, e.g. the case and the lemma of the preposition for nouns; see Žabokrtský et al. (2008),
- **sempos** - semantic part of speech.

The last three factors are defined only for content words, which have an explicit representation on the tectogrammatical (deep syntactic) layer. The remaining words are assigned blank values.

We define two types of collocation features on the surface form and the above mentioned factors:

- exact collocation and
- loose collocation.

[2] We use the same annotated data as in Bojar and Kos (2010).

### Exact Collocation

Under exact collocation we understand the same type of collocation as Gimpel and Smith (2008) used in their work. For the phrase $f_k^l$, which begins at position $k$ and ends at position $l$, we define the contextual probability

$$pExact_{factor}(e|factor_{k-left}^{k-1}, f_k^l, factor_{l+1}^{l+right}) \tag{4.11}$$

where $e$ is the target phrase, $left$ is the size of the left context and $right$ the size of the right context of the phrase $f_k^l$; $factor_{k-left}^{k-1}$ and $factor_{l+1}^{l+right}$ are sequences of factors of the type $factor$ surrounding the phrase $f_k^l$. It is possible to define different lengths of the left and right context; context of zero length means that it is not considered at all. The contextual probability is estimated from frequency counts from the training data and it requires that the context is exactly matched when extracting the counts. Figure 4.3 shows an example of an exact collocation feature on POS tags with the left context of two factors and the right context of one factor.

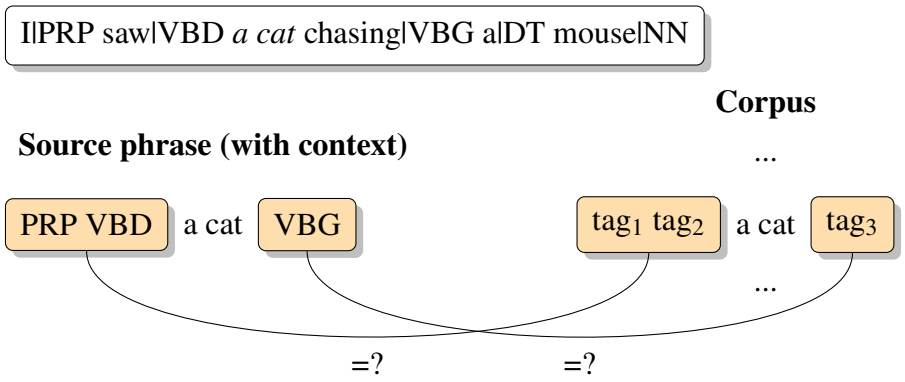



Figure 4.3: Exact collocation features require that the left and the right context of a phrase match in the given factor. The count of phrases with matching context is extracted from the corpus and used to compute the feature value.

### Loose Collocation

While exact collocation requires exact match of the left and right context, loose collocation views surrounding context as a set of factors and computes how many factors from the context are confirmed by the training data. The number of confirmed factors is then normalized by the size of the context. The loose contextual probability is formally defined as

$$pLoose_{factor}(e|factor_{k-left}^{k-1}, f_k^l, factor_{l+1}^{l+right}) = \frac{\sum_T \frac{|Context(f_k^l) \cap Context(T_i)|}{|Context(f_k^l)|}}{|T|} \tag{4.12}$$

where $T$ is the set of occurrences of the phrase pair $\langle f_k^l, e \rangle$ in the training data; $Context(f_k^l)$ is a set of factors from the left and right context of the source phrase $f_k^l$; $Context(T_i)$ is a set of factors from the left and right source context of the $i$-th occurrence of the phrase pair $\langle f_k^l, e \rangle$ in the training data. Figure 4.4 shows and example of a loose collocation feature with the left and right context of three factors.

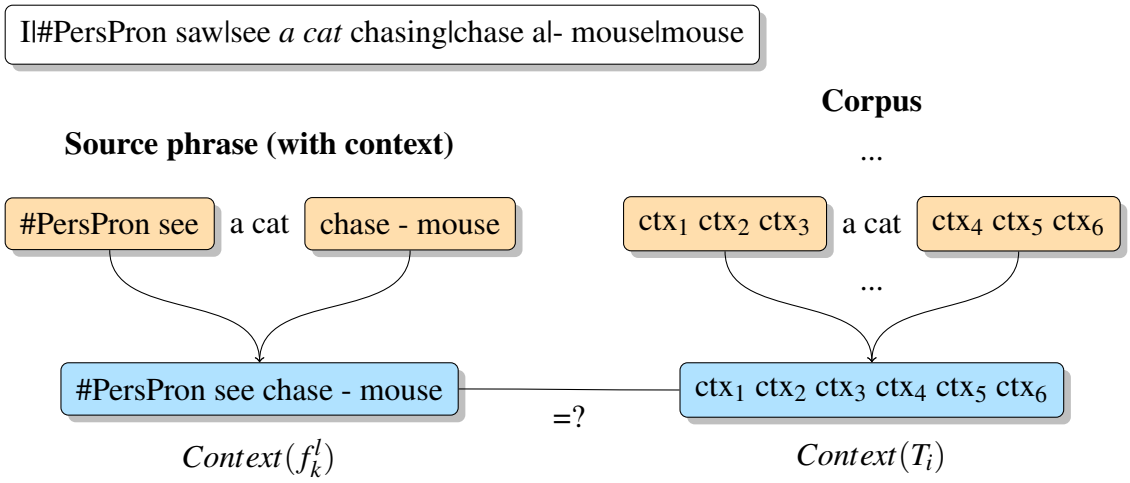



Figure 4.4: Loose collocation features create a set of context factors $Context(f_k^l)$ and $Context(T_i)$ first (same tokens are used only once). Then, the number of matching factors is counted.

The context cannot cross sentence boundaries for exact nor loose collocation, i.e. the left context of a word at the beginning of a sentence is always empty and the right context for the last word in a sentence as well. Alternatively, we could use special values to pad the beginning and the end of the sentence.

### 4.3.2 Dependency Features

Dependency features require that the source sentence is parsed by a syntactical parser. We do not use constituent parse tree as Gimpel and Smith (2008), but dependency tree instead. Therefore we need to define dependency features on the dependency parse tree structure. We examine three features:

- **depth of the phrase** - lowest depth of a word from the phrase in the dependency parse tree,
- **is the phrase part of a subtree** - is it (fully) connected by the parent-child relation,
- **position relative to the sentence root** - is the phrase to the right, to the left or does it contain the root word.

### 4.3.3 Logistic Function

Although it would be possible to use the source-context feature values directly in the log-linear model, we pass their logarithm to the model instead. Since the above mentioned source-context features return a score from the interval $[0,\infty)$, we need to be careful if they returned zero because the logarithmic function is defined only on the interval $(0,\infty)$. Therefore, we use the logistic function

$$f_{logistic}(x) = \frac{1}{1+exp(-x)} \tag{4.13}$$

to transform the feature values first and then apply the logarithm. The logistic function is defined on real numbers and returns a value from the interval $(0,1)$. The values passed to the log-linear model undergo to following transformation

$$log(f_{logistic}(feature\_value)) \tag{4.14}$$

and are always negative. We see an advantage in this approach because the transformed feature values are from a clearly defined range and the model and especially the parameter optimization algorithm MERT are not confused by arbitrarily large numbers.

## 4.4 Implementation Details

In this section we describe the implementation of the source-context model and its integration in the phrase-based MT system Moses.

Since we would like to compute the source-context features over a richly annotated corpus, which can itself consume a lot of memory, it would be intractable to store the feature values for each translation pair explicitly. Instead, we use a data structure that can efficiently compute the frequency and location of the translation phrase pairs in large bilingual corpora, and estimate the feature values on-the-fly.

### 4.4.1 Suffix Arrays

We use a special data structure called suffix array (Manber and Myers; 1990) to compute the necessary document statistics. This data structure creates an index for searching substrings or n-grams in a large corpus. Its biggest advantage is that it requires only $O(n)$ space for its own internal representation, where $n$ is the size of the corpus, but it allows to compute frequency counts of an arbitrary phrase only in $O(log(n))$ time.

Abstractly, a suffix array is a lexicographically-sorted list of all suffixes in a corpus, where a suffix is a substring running from each position in the text to the end. However, rather than actually

storing all suffixes, a suffix array can be constructed by creating a list of references to each of the suffixes in a corpus. Figure 4.5 shows how a suffix array is initialized for a corpus containing the sentence *banana* (for the sake of simplicity we represent words as single characters, i.e. *banana* illustrates a sentence containing six words). Each token in the corpus has a corresponding place in the suffix array, which is identical in length to the corpus. The final state of the suffix array is shown on the right side of Figure 4.5, which is as a list of the indices of tokens in the corpus that corresponds to a lexicographically sorted list of the suffixes. The advantages of this representation are that it is compact and easily searchable. Typically, it is stored as an array of integers where the array has the same length as the corpus. Because it is organized lexicographically, any phrase can be quickly located within it using binary search.

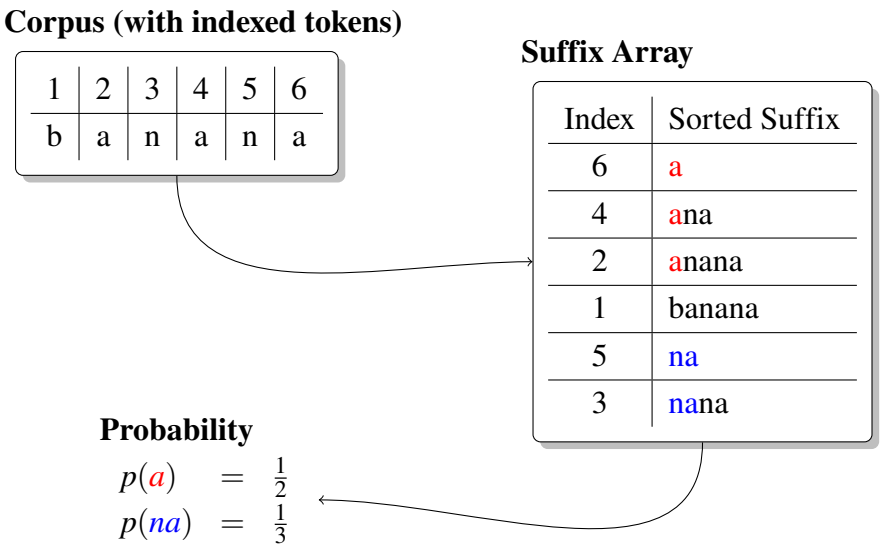



Figure 4.5: Suffix arrays are created by sorting word indexes from the corpus. N-gram positions in the corpus can be found by two binary searches which locate the beginning and the end of the n-gram in the suffix array.

### Bilingual Suffix Arrays

Suffix arrays need to be adapted for statistical MT so that they can operate over bilingual corpus (Callison-Burch et al.; 2005). The following components are required:

- Two suffix arrays: one created from the source language part of the corpus, the other one created from the target language part of the corpus.

- An index that tells us the correspondence between sentence numbers and positions in the source and target language corpora.
- A word alignment for each sentence pair in the parallel corpus.

In order to compute the source-context feature values, we need to have a method to locate positions of the source phrase $f$ in the source corpus given a source-target translation pair $\langle f, e \rangle$. This method should be fast to compute because we do not want to spoil the memory advantage achieved by using suffix arrays. The position extraction method consists of two phases:

1. A set $S$ of source phrase positions in the source corpus, and a set $T$ of target phrase positions in the target corpus are extracted. The extractions take $O(2log(n))$ time, where $n$ is the size of the corpus, since they require two binary searches to locate the beginning and end of the source/target phrase in the respective corpus.
2. The smaller set of positions is taken (either $S$ or $T$) and aligned phrases from the complemental part of the corpus are extracted for each position using the word alignment. If the aligned phrase matches the respective part of the translation pair, the source corpus position of the source phrase is stored in a set $A$ of aligned translation pairs. After the second phase, the set $A$ contains positions in the source corpus at which the source phrase $f$ begins, which is aligned to the target phrase $e$.

This approach is very efficient if at least one of the phrases $e$ or $f$ occurs only a few times in the corpus. However, we also need to handle situations in which both the source and the target phrase are very frequent in the corpus, e.g. if the phrase pair consists of full stop on both sides. Since full stop appears (almost) in every sentence, we would have to evaluate every feature on millions of occurrences if we took the Large dataset (which contains more than 7M sentences).

To solve this problem, we select only a small sample of the maximum size $M$ from all occurrences in the set $A$. This does not influence phrase pairs which occur rarely because they fit into the limit completely, but reduces large sets to a reasonable size. Callison-Burch et al. (2005) showed that sample size of 500 elements should be sufficient to estimate translation probabilities from bilingual suffix arrays. We use the same threshold to limit the size of the set $A$.

Nevertheless, we also need to limit the size of the sets $S$ and $T$, which contain positions of the source/target phrase in the source/target corpus, because we use the smaller one of them to create the set $A$. Their size should be larger than $M$ (maximum size of $A$) because we discard some occurrences that are not aligned to a phase which matches the other part of the phrase pair. We limit the size of $S$ and $T$ to be at most five times larger than $A$, i.e. at most 2500 elements. This limit is arbitrarily chosen and a more detailed analysis should be done to find the optimal value.

Lopez (2007) suggests a different solution to the problem of too frequent phrases. Because the number of very frequent phrases is (due to Zipf's Law) relatively small they precompute their positions in the corpus and store them in a separate lookup table. Nevertheless, we follow the subsampling approach because it does not require any special precomputation and is easier to implement.

#### Corpus with Restricted Factors

Source-context features work with rich annotation of data. Each word from the source sentence, i.e. the sentence being translated, and the source language training corpus must contain rich factors which the source-context features work with. When we create the suffix arrays we need to sort the word indexes according to the lexicographical ordering defined on the words. However, using a richly annotated corpus would shuffle the preferred lexicographical ordering on surface forms. For example the POS tags used as a rich factor would force the suffix array to sort the array according to the current POS tag first and not according to the following word form. Thus, we would not be able to extract long n-grams correctly. Therefore, we need to create two parallel corpora for each suffix array:

1. corpus with full annotation,
2. corpus with restricted factors over which we will create the suffix array. Usually, this corpus will contain only surface forms to represent plain n-grams.

After the suffix array is created, i.e. the indexes are sorted according to the lexicographical ordering on the restricted factors, we can lookup the richly annotated words in the full corpus using word indexes stored in the suffix array. We can do that because the positions of words in the restricted corpus are the same as in the full corpus. When we extract a word with the index $i$ from the suffix array, we can directly look at the $i$-th position in the full corpus to get the rich factors of the word. The restricted corpus is used only for the creation of the suffix array.

### 4.4.2 Integration into Moses

In our work, we took advantage of an already existing implementation of suffix arrays in Moses. However, it was necessary to extended it so that it can work with rich factors, and adapt it for the extraction of bilingual phrases.

The interface between Moses and the context features is the class `SourceContextFeatures`, an implementation of the abstract class `StatefulFeatureFunction`. The concrete class contains a pointer to the class `SourceContextBilingualDynSuffixArray` which stores pointers to the individual source-context feature classes (see Figure 4.6).

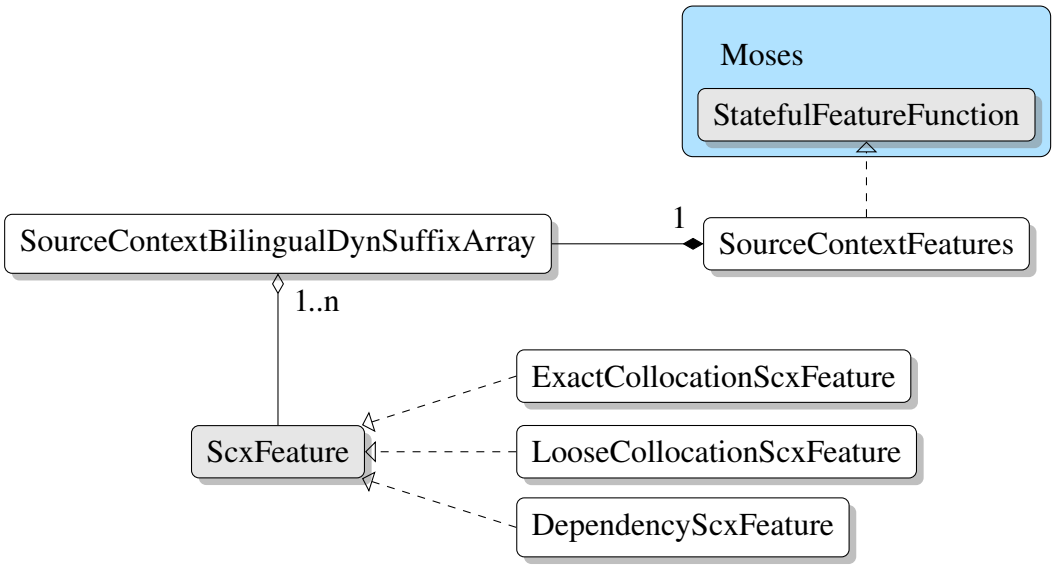



Figure 4.6: A simplified UML diagram for classes related to source-context features.

When Moses collects feature scores for a translation option the function

```
virtual FFState* Evaluate(
            const Hypothesis& cur_hypo,
            const FFState* prev_state,
            ScoreComponentCollection* accumulator)
```

declared in the abstract class `StatefulFeatureFunction` is called. Because a translation option of a source phrase can occur at many different places in the target sentence, the function can be called many times with similar parameters that differ only in the position of the target phrase in the target sentence. Since this difference does not have any influence on the source-context scores, we can cache the already computed values and reuse them later within the same source sentence. This can significantly speed up the computation time.

### Aggregation of Feature Scores

With the increasing number of source-context features the MERT algorithm needs to optimize more and more feature weights. This negatively influences the runtime of the algorithm and makes it more difficult to find optimal weights. Birch and Osborne (2007) suggested that using one cumulative score instead of many less informative ones could have better impact on the translation quality. Therefore, we implemented the option to aggregate all features into one cumulative score which is passed to Moses. In the MERT algorithm, only one weight for the aggregate score needs to be optimized.

We allow to assign weights to the features which are aggregated. Thus, the aggregate score is a linear combination of the individual feature scores. The weights need to be optimized by an external program. We do no provide any method to optimize them automatically.

### Moses Configuration

There are two possible ways to pass parameters to Moses. Either through command line arguments or in a configuration file, usually called *moses.ini*. Source-context features can be configured in both ways. We will describe only the latter one because the command line arguments require the same format as the configuration file (for more details see the Moses documentation[3]).

There are two parameters that define which source-context features will be used:

- `source-context-file`
  This parameter consists of six parts which are written on the same line separated by a blank space:

  1. Factor specification for the richly annotated corpus, e.g. `'0,1,2-0'` which means that the parallel corpus has three input (source) factors and one output (target) factor. The factor indexes must match indexes used in other parts of the Moses configuration file.
  2. Factor specification for the restricted corpus, e.g. `'0-0'`. In this case, only the first factor on the source and the target side will be used to create (sort) the suffix array.
  3. Number of weights passed to the log-linear model.
  4. Specification of the source-context features. Individual features are separated by `'|'`. The exact form of the specification is described below.
  5. Aggregate weights separated by `'|'`, or the string `none` if all feature scores are passed to the log-linear model. The number of aggregate weights must match the number of features.
  6. Path to the richly annotated bilingual corpus. The files `<path_to_corpus>.src`, `<path_to_corpus>.trg` and `<path_to_corpus>.ali` must exist, which contain the source corpus, the target corpus and the alignment file. The source and target corpus must contain each sentence on a separate line, and the alignment file must be in the standard alignment format used by Moses (e.g. `'0-1 1-2 2-0'`). The direction of the alignment pairs should be source→target word index.

[3] http://www.statmt.org/moses/

- `weight-sc`
  This parameter is followed by the log-linear model weights for the source-context features, each weight on a separate line. The number of weights must match the number of features. If aggregate weights were specified in the parameter `source-context-file`, only one weight should be used.

Figure 4.7 shows an excerpt from a Moses configuration file which specifies two source-context features.

```
# source-context features specification
[source-context-file]
0,1,2-0 0-0 2 e:1:-1|l:2:-5+5 none /home/kos/corpora/corpus-news

# source-context weights
[weight-sc]
0.5
0.5
```

Figure 4.7: Moses configuration file excerpt specifying two source context features: one *exact collocation* feature using the directly preceding factor with index 1, and one *loose collocation* feature using five preceding and five following factors with index 2 as context.

Now, we will describe how the features are specified. Exact and loose collocation features follow the same specification pattern:

`<type>:<factor>:<context>` (4.15)

where `<type>` is the type of the feature: the string `'exact'` or `'e'` for exact collocation, and `'loose'` or `'l'` for loose collocation; `<factor>` is the index of the source factor the collocation feature works on, and `<context>` is the size of the left and right context in the form `'-<left>+<right>'`, where `<left>` and `<right>` are a single digit.

For exact collocation, it is possible to specify the zero context `'-0+0'`. This is an extension which does not follow the formal definition of the exact collocation. In this case we do not check the surrounding context but all factors *inside* of the current phrase. This option can be viewed as an alternative to factored translation because we condition the translation options on the factors of the source phrase. The zero context cannot be used for loose collocation.

Dependency features follow a similar specification pattern as collocation features:

`<type>:<dep_factor>:<subtype>` (4.16)

where <type> is the string 'dependency' or 'd'; <dep_factor> is the index of the source factor that contains index of the (dependency-tree) parent in the sentence; and <subtype> specifies the subtype of the dependency feature, as they were introduced in Section 4.3.2:

- 0 - depth in the dependency tree,
- 1 - subtree in the dependency tree,
- 2 - phrase position relative to the root word.

## 4.5 Experiments

In order to test whether the implemented source-context features can improve MT translation quality, we run several experiments. In all experiments we optimize the feature weights towards BLEU using Z-MERT as in Chapter 3. We use the script *zmert-moses.pl* to launch Z-MERT.

### 4.5.1 Training Data

There are four possible combinations of the Small and the Large data (which were introduced in Section 3.3.1) to train the simple phrase-based system and the source-context model:[4]

1. Small phrase-based + Small source context,
2. Large phrase-based + Small source context,
3. Small phrase-based + Large source context,
4. Large phrase-based + Large source context.

The configuration of the simple phrase-based system, which we denote as baseline in the following experiments, is the same as in Section 3.3.1.

We encountered a performance problem when we tried to run the source-context model on the Large data. For each MERT iteration the suffix arrays need to be created. While the creation of suffix arrays takes less than minute for the Small data (126k sentences), we need more than half an hour only for parsing the suffix array corpora for the Large data (7.5M sentences). This makes the MERT optimization too slow. Figure 4.8 gives more details on the time necessary to load the suffix arrays. We also include the optimized BLEU and SemPOS scores for both experiments,

[4]The term "training" is not exact for the source-context model because there is no training phase. We use the "training" data to construct the suffix arrays and compute the feature values on-demand during decoding. Nevertheless, we use this term to call the data in a unified way.

which used Small data to train the baseline and five loose collocation features on surface forms as the source-context model.

| Data | Sentences | Time to Load | BLEU | SemPOS | MERT Runtime |
|---|---|---|---|---|---|
| Small | 126,144 | 46s $<$ 1m | 10.42±0.42 | 29.89 | 4h24m (8) |
| Large | 7,544,465 | 2712s $\sim$ 45m | 10.23±0.42 | 29.59 | 1d22h29m (20) |

Figure 4.8: Initialization of suffix arrays (Time to Load) takes too long for the Large training data. Moreover, the Large data did not bring an improvement in BLEU nor in SemPOS score (Small data used for baseline). Number of MERT iterations in brackets.

There is no improvement in neither of the scores for the Large data compared to the Small data used in the source-context model. Therefore, we run our experiments only for the first two combinations, i.e. Small simple phrase-based + Small source context and Large simple phrase-based + Small source context, which use only the Small data for the source-context model. In the following text we will denote the first option as Small (using Small data to train the baseline system), and the second option as Large (using Large data to train the baseline system).

### 4.5.2 Collocation Feature Sets

Because of the large number of possible combinations of the collocation features, we group them into the so-called *elementary sets*. These sets are parametrized by the factor they try to match: we use 1) surface form, 2) POS tag, 3) lemma, 4) formeme and 5) sempos as factors. Each elementary set contains five feature functions with different context sizes:

- Exact collocation:
  - left = 1, right = 0,
  - left = 0, right = 1,
  - left = 2, right = 0,
  - left = 0, right = 2,
  - left = 1, right = 1.
- Loose collocation:
  - left = 3, right = 0,
  - left = 0, right = 3,

- left = 5, right = 0,
- left = 0, right = 5,
- left = 3, right = 3.

Thus, we have five elementary sets, which we will denote *form*, *pos*, *lemma*, *formeme* and *sempos*, for both collocation types, and each elementary set contains five source-context features.

We do not need to define elementary sets for dependency features because there are only three of them.

### 4.5.3 Aggregated Features

We are also interested in the influence of aggregating all source-context features into one cumulative score. We did not implement any special algorithm to find the optimal weights for the linear combination of individual features into the cumulative score. We simply take the weights that were found by MERT when we evaluated the elementary sets and used all source-context feature values in the log-linear model. This is a very coarse approach because the weights were optimized together with other features of the log-linear model. Therefore they can be strongly influenced by other components of the log-linear model, which probably have bigger influence on the translation quality than the source-context features.

## 4.6 Results

The obtained results do not provide any conclusive evidence that source-context features can significantly improve translation quality. Even though slight improvement can be observed for selected features, it is only small and within the confidence intervals for BLEU.

#### Small Data

First, we took all features of the same type and used them in one source-context model: there were 25 exact collocation features (five factors $\times$ five context sizes), 25 loose collocation features and 3 dependency features. Furthermore, we combined all features into one big model, which had 53 features all together. Figure 4.9 shows BLEU and SemPOS scores for these models measured on the Small data.

The biggest improvement both in BLEU and SemPOS scores was achieved by the dependency features. We can also see that there is negative correlation between the number of features used in the model and the BLEU score. This implies that if the number of features becomes too big, MERT has difficulties to find the optimal weights.

| | Scores | | MERT | |
|---|---|---|---|---|
| | BLEU | SemPOS | Time | Iters |
| Baseline | 10.21±0.39 | 29.65 | 4h55m | 18 |
| All Exact (25) | 10.39±0.43 | 29.56 | 12h46m | 12 |
| All Loose (25) | 10.33±0.44 | 29.68 | 21h12m | 18 |
| All Dep. (3) | 10.43±0.42 | 29.81 | 10h58m | 15 |
| All (53) | 9.93±0.43 | 29.73 | 2d00h54m | 20 |

Figure 4.9: BLEU and SemPOS scores for the baseline and the combined source-context features (number of individual features in brackets) - Small data.

To get more detailed information about the performance of the collocation features, we also ran experiments with the elementary feature sets described in Section 4.5.2. Figure 4.10 shows that the highest SemPOS score was achieved by exact collocation features on POS tags. The highest score for loose collocation features was on surface forms, closely followed by lemmas. We explain this that part-of-speech tags can help to disambiguate neighbouring words in exact collocation, whereas surface forms and lemmas can have impact over long distances in loose collocation. On the other hand, the performance of POS tags in loose collocation is close to the baseline which suggests that they are less effective at long distances. The BLEU scores for individual elementary sets do not differ very much but they usually correlate with the SemPOS scores.

| | Exact | | Loose | |
|---|---|---|---|---|
| Factor | BLEU | SemPOS | BLEU | SemPOS |
| Baseline | 10.21±0.39 | 29.65 | 10.21±0.39 | 29.65 |
| form | 10.37±0.42 | 29.57 | 10.42±0.42 | 29.89 |
| pos | 10.39±0.41 | 30.14 | 10.29±0.43 | 29.65 |
| lemma | 10.30±0.43 | 29.82 | 10.35±0.43 | 29.83 |
| formeme | 10.34±0.41 | 29.83 | 10.31±0.42 | 29.68 |
| sempos | 10.42±0.42 | 29.79 | 10.40±0.41 | 29.62 |

Figure 4.10: Detailed BLEU and SemPOS scores for exact and loose collocation features - Small data.

### Large Data

We ran the same experiments for the Large data as for the Small data. Figure 4.11 contains results for the grouped source-context models, which used 25 exact collocation features, 25 loose collocation features, 3 dependency features and a combination of all 53 source-context features. The BLEU score was consistently better for all four source-context models than for the baseline system but within the confidence intervals. On the contrary, SemPOS score was the highest for the baseline system. If we compare it to the Small data results, where the SemPOS score increased in almost all cases, we can argue that source-context features helped in the Small data setting with better lexical choice. The impact was smaller for the Large data, because the translation model alone could provide better lexical choice. The drop in the metric scores for source-context models containing large number of features in not as high as for the Small data.

| | Scores | | MERT | |
|---|---|---|---|---|
| | BLEU | SemPOS | Time | Iters |
| Baseline | 12.78±0.45 | 31.45 | 1h48m | 4 |
| All Exact (25) | 12.89±0.46 | 31.42 | 9h57m | 6 |
| All Loose (25) | 12.81±0.46 | 31.02 | 10h20m | 6 |
| All Dep. (3) | 12.97±0.45 | 31.15 | 4h58m | 6 |
| All (53) | 12.88±0.46 | 31.38 | 2d09h18m | 9 |

Figure 4.11: BLEU and SemPOS scores for the baseline and the combined source-context features (number of individual features in brackets) - Large data.

| | Exact | | Loose | |
|---|---|---|---|---|
| Factor | BLEU | SemPOS | BLEU | SemPOS |
| Baseline | 12.78±0.45 | 31.45 | 12.78±0.45 | 31.45 |
| form | 12.91±0.46 | 31.43 | 12.92±0.46 | 31.47 |
| pos | 12.95±0.45 | 31.09 | 12.86±0.47 | 31.51 |
| lemma | 12.82±0.47 | 31.56 | 12.90±0.46 | 31.46 |
| formeme | 12.79±0.46 | 31.33 | 12.76±0.44 | 31.32 |
| sempos | 12.81±0.46 | 31.24 | 12.76±0.45 | 31.30 |

Figure 4.12: Detailed scores for exact and loose collocation features - Large data.

The detailed scores for collocation features in Figure 4.12 confirm our observations for Small data: exact collocation features appreciate morphological advice and perform the best on part-of-

speech tags, whereas loose collocation benefits the most from surface forms and lemmas, which supply information about lexical choice.

### Aggregate Features

Aggregate features, which pass only one score (a linear combination of source-context feature scores) to the log-linear model, did not bring any significant improvement. We evaluated the aggregate features on elementary sets that had the most promising results in the previous section—pos and sempos for exact collocation, and form and lemma for loose collocation. The evaluation was done only on the Small data.

Figure 4.13 shows that the BLEU scores did not change much compared to the baseline but there is a consistent decrease in SemPOS for all experiments. We denote the experiments from which the aggregate weights were taken as baseline.

| | | Baseline | | Aggregate | |
|---|---|---|---|---|---|
| Colloc. | Factor | BLEU | SemPOS | BLEU | SemPOS |
| Exact | pos | 10.39±0.41 | 30.14 | 10.38±0.42 | 29.43 |
| Exact | sempos | 10.42±0.42 | 29.79 | 10.32±0.42 | 29.59 |
| Loose | form | 10.42±0.42 | 29.89 | 10.43±0.43 | .29.41 |
| Loose | lemma | 10.35±0.43 | 29.83 | 10.42±0.43 | 29.61 |

Figure 4.13: Aggregate features do not help to improve translation quality, probably due to bad-quality aggregate weights - Small data.

### Zero and Wide Context

We also wanted to see the influence of zero context for exact collocation and wide context for loose collocation features. Zero context (`'-0+0'`) is defined only for exact collocation features and means that instead of looking on surrounding words, we require that factors of words inside of the phrase match the source phrase factors (see Section 4.4.2). We evaluate zero context for pos and formeme factors.

We consider five preceding and five following words `'-5+5'`, and nine preceding words and nine following words `'-9+9'` of a phrase as wide context for loose collocation features, which we evaluate only on lemma that showed the most promising results on elementary sets.

Figure 4.14 shows metric scores for selected exact and loose collocation features. In all experiments, the source-context model consists of only one feature. We evaluate the features on

the Small data and the baseline scores are reported for Moses configuration that does not use the source-context model.

| Colloc | Context | Factor | BLEU | SemPOS |
|---|---|---|---|---|
| Baseline | - | - | 10.21±0.39 | 29.65 |
| Exact | -0+0 | pos | 10.33±0.43 | 29.63 |
| Exact | -0+0 | formeme | 10.14±0.41 | 29.71 |
| Loose | -5+5 | lemma | 10.09±0.44 | 29.76 |
| Loose | -9+9 | lemma | 10.44±0.41 | 29.85 |

Figure 4.14: Zero and wide context - Small data.

The exact collocation features show mixed results for BLEU and SemPOS. There is an improvement in SemPOS for the loose collocation features which is consistent with observations for other loose collocation features. This suggests that loose collocation on lemma can help with better lexical choice which is appreciated by SemPOS. Nevertheless, the BLEU score for collocation features oscillates around the baseline score.

# Chapter 5

# Discussion and Future Work

Neither of the two approaches we tried brought statistically significant improvement in translation quality. While using SemPOS as optimization metric in MERT showed to be unstable and influenced the MT quality rather negatively, there were several promising results for the source-context model. Nevertheless, our experiments did not provide any conclusive evidence about statistically significant improvement.

### SemPOS in MERT

The main reason why SemPOS is not a suitable MERT optimization metric is that it completely ignores surface forms and focuses only on matching correct lemmas. While this is sufficient when evaluating output of systems which already tried to target the correct surface forms, it is not suitable for defining the MERT error space. The SemPOS score cannot distinguish between two sentences that contain words having the same lemmas but different word forms in different word order. This forces the MERT algorithm to make an arbitrary choice which translation is better.

The linear combination of SemPOS and BLEU results in increased stability of the optimized weights. Nevertheless, it would be advisable to get some human feedback in form of manual evaluation of the system output whether the combination of both metrics resulted in increased translation quality.

### Source-Context Model

The contribution of the source-context model to better translation quality could not be proved. There are many factors that could influence the performance of the source-context features:

- **Size of the source-context corpus**
  We could compute the source-context features only on the Small data because of the ex-

tremely long loading time of suffix arrays. The only experiment we did using the Large data for the source-context model did not show any quality improvement in MT metric scores. However, the lower performance could be only on the selected features and some other features could have better impact on the translation quality. After modification of the loading procedure for suffix arrays, it would be interesting to evaluate the features also on the Large data. There is a large potential for improvement: many phrase pairs, which did not occur in the Small data, received the default score. By evaluating the features on the Large data we could get feature values with higher discriminative power.

- **Better sampling method**
  Before evaluating the source-context features for a given source-target phrase pair, we need to extract their positions in the corpus. The implemented method can discard some phrase occurrences if they do not fit into the sample limit. This can be a problem if the source phrase occurs many times in the source corpus, the target phrase occurs many times in the target corpus, but they are aligned to each other only in few cases. When we create the samples from either side of the corpus we can discard by chance the few aligned occurrences. Therefore, a more fine-grained approach could be used that would first filter the sentences (i.e. sentence pairs from the bilingual corpus) in which both the source phrase and the target phrase occur. However, we need to design this filtering method with computational complexity in mind because it would be intractable to go through all sentences in the corpus for large corpora containing millions of sentences.

- **Additional features**
  We evaluated only a limited set of source-context features. The source-context model is flexible enough to compute additional features that use other linguistically motivated factors. It is sufficient to provide the new factors in the source corpus. Another approach would be to include in the collocation context only words that fulfill some condition, e.g. take as context only content words. Similar approach was described by Chan et al. (2007) who used only nouns, verbs and adjectives as context. Furthermore, the dependency features could be enhanced by features that check that the parent or sons' factors match a given value.

- **Optimization of aggregate weights**
  We tried only a rough estimation of the aggregate feature weights, which we obtained from the MERT optimization. Some more sophisticated method should be used to obtain better weights that would better reflect the contribution of the individual features to the aggregate score.

To illustrate the performance of individual source-context features, we selected two example sentences from the test set and show the baseline output together with the output of exact collocation features on part-of-speech tags (pos) and loose collocation features on surface forms (form) for the Large data.

Figure 5.1 shows a fragment of a sentence for which the source-context features provided a better translation than the baseline system without the source-context model. Both of the features selected the correct word forms for the translation of the phrase *chief economist*. The baseline system selects correct words but puts them into nominative, even though they should be in genitive, which was correctly recognized by both source-context features. On the other hand, exact collocation features make a mistake when translating the phrase *proposed plan*. They choose a translation option *návrh plánu* that is grammatically correct but has a slightly different meaning than the reference translation.

| | |
|---|---|
| SRC | According to the chief economist of Patria Finance , David Marek , the proposed plan is a good idea ... |
| REF | Navrhovaný plán je podle hlavního ekonoma Patria Finance Davida Marka dobrým nápadem ... |
| Baseline | Podle hlavní$_{nom}$ ekonom$_{nom}$ Patria Finance , David Marek , navrhovaný$_{adj}$ plán$_{noun}$ je dobrý nápad ... |
| Exact:pos | Podle hlavního$_{gen}$ ekonoma$_{gen}$ , David Marek z Patria Finance , návrh$_{noun}$ plánu$_{noun}$ je dobrý nápad ... |
| Loose:form | Podle hlavního$_{gen}$ ekonoma$_{gen}$ , David Marek z Patria Finance , navrhovaný$_{adj}$ plán$_{noun}$ je dobrý nápad ... |

Figure 5.1: Source-context features can help to choose better word forms (underlined) than the baseline system. However, they can also force the system into an unsuitable translation, that conveys a different meaning than the original sentence (framed).

| | |
|---|---|
| SRC | Then I scurried to my seat . |
| REF | Pak jsem rychle vplula na své místo . |
| Baseline | Pak jsem se na mém místě$_{loc}$ . |
| Exact:pos | Pak jsem se na své místo$_{acc}$ . |
| Loose:form | Pak jsem se na moje místo$_{acc}$ . |

Figure 5.2: Both source context features choose a translation option in correct case. Moreover, the exact collocation features managed to recognize that they need to use the possessive reflexive pronoun *své* instead of the possessive pronoun *moje*.

The second example sentence, depicted in Figure 5.2, shows that the collocation features can help with case selection. If we omit the mistake of leaving out the translation of the verb *scurried*[1] which is done by all three systems, both collocation features correctly choose the translation of *seat* in accusative (*místo*) whereas the baseline system prefers locative (*místě*). Moreover, exact

[1] Probably due to the fact that this word is not in the dictionary of the translation system.

collocation features choose the correct form of the possessive pronoun *my→své*. The baseline system and the loose collocation features select the wrong word form *mém* and *moje*.

# Chapter 6

# Conclusion

In this thesis we investigated two possible ways to improve translation quality of statistical phrase-based MT systems.

First, we adapted the MERT algorithm to be able to optimize the log-linear model parameters towards a metric using rich annotation—SemPOS. Although the metric is good at comparing different MT systems, the results show that it is bad at comparing candidates from a single system in an n-best list. Therefore, the optimized parameters strongly vary in quality measured in BLEU and SemPOS score.

Second, we implemented the source-context model based on suffix arrays. The biggest advantage of suffix arrays is that they can effectively represent large collections of richly annotated data and provide methods for fast extraction of n-grams from the data. Richly annotated bilingual copora can be used to define a wide range of features that are directly used in the log-linear model. We introduced a set of collocation and dependency features on the source side over factors containing rich linguistic information. The feature values are calculated on-demand during decoding and there is no need for training. Thus, the source-context model is easy to use and fast to deploy.

We evaluated the contribution of the source-context model only on small data because we encountered performance problems connected with loading of the suffix arrays. This problem could be solved by loading the suffix arrays from a precomputed format in which all words are represented by an integer value instead of the string. The highest MT quality improvement was measured for exact collocation on POS tags, and loose collocation on surface forms and lemmas. However, the improvement was only moderate and within confidence intervals of BLEU. Therefore, it is not possible to state that the source-context model improves translation quality.

# Bibliography

Banerjee, S. and Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments, *Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43th Annual Meeting of the Association of Computational Linguistics (ACL-2005)*, Ann Arbor, Michigan, pp. 65–72.

Birch, A. and Osborne, M. (2007). CCG Supertags in Factored Statistical Machine Translation, *Proceedings of the Second Workshop on Statistical Machine Translation*, Association for Computational Linguistics, pp. 9–16.

Bojar, O. and Kos, K. (2010). 2010 Failures in English-Czech Phrase-Based MT, *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, Association for Computational Linguistics, Uppsala, Sweden, pp. 60–66.
**URL:** *http://www.aclweb.org/anthology/W10-1705*

Bojar, O., Kos, K. and Mareček, D. (2010). Tackling Sparse Data Issue in Machine Translation Evaluation, *Proceedings of the Association for Computational Linguistics*, Uppsala, Sweden.

Bojar, O. and Žabokrtský, Z. (2009). CzEng 0.9: Large Parallel Treebank with Rich Annotation, *Prague Bulletin of Mathematical Linguistics* **92**: 63–83.

Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L. and Roossin, P. S. (1990). A Statistical Approach to Machine Translation, *Computational Linguistics* **16**(2): 79–85.

Callison-Burch, C., Bannard, C. and Schroeder, J. (2005). Scaling Phrase-Based Statistical Machine Translation to Larger Corpora and Longer Phrases, *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 255–262.

Callison-Burch, C., Fordyce, C., Koehn, P., Monz, C. and Schroeder, J. (2008). Further Meta-Evaluation of Machine Translation, *Proceedings of the Third Workshop on Statistical Machine*

*Translation*, Association for Computational Linguistics, Columbus, Ohio, pp. 70–106.
**URL:** *http://www.aclweb.org/anthology/W/W08/W08-0309*

Callison-Burch, C., Koehn, P., Monz, C., Peterson, K., Przybocki, M. and Zaidan, O. (2010). Findings of the 2010 Joint Workshop on Statistical Machine Translation and Metrics for Machine Translation, *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, Association for Computational Linguistics, Uppsala, Sweden, pp. 17–53.
**URL:** *http://www.aclweb.org/anthology/W10-1703*

Callison-Burch, C., Koehn, P., Monz, C. and Schroeder, J. (2009). Findings of the 2009 Workshop on Statistical Machine Translation, *Proceedings of the Fourth Workshop on Statistical Machine Translation*, Association for Computational Linguistics, Athens, Greece, pp. 1–28.
**URL:** *http://www.aclweb.org/anthology/W/W09/W09-0401*

Carpuat, M. and Wu, D. (2005). Word Sense Disambiguation vs. Statistical Machine Translation, *Proceedings of the 43rd Annual Meeting of the ACL*, Association for Computational Linguistics, Ann Arbor, pp. 387–394.

Carpuat, M. and Wu, D. (2007). Context-Dependent Phrasal Translation Lexicons for Statistical Machine Translation, *Proceedings of Machine Translation Summit XI*, Copenhagen, Denmark, pp. 73–80.

Carpuat, M. and Wu, D. (2008). Evaluation of Context-Dependent Phrasal Translation Lexicons for Statistical Machine Translation, *Sixth International Conference on Language Resources and Evaluation*, Marrakech, Morocco.

Chan, Y. S., Ng, H. T. and Chiang, D. (2007). Word Sense Disambiguation Improves Statistical Machine Translation, *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 33–40.

Chandioux, J. and Guraud, M.-F. (1981). Météo: Un Système à l'Épreuve du Temps, *Méta* **26**(1): 18–22.

Chiang, D. (2005). A Hierarchical Phrase-Based Model for Statistical Machine Translation, *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 263–270.

Chiang, D. (2007). Hierarchical Phrase-Based Translation, *Computational Linguistics* **33**(2): 201–228.

Doddington, G. (2002). Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics, *Proceedings of the Second International Conference on Human Language Technology Research*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 138–145.

Eisner, J. (2003). Learning Non-Isomorphic Tree Mappings for Machine Translation, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, pp. 205–208.

Gildea, D. (2003). Loosely Tree-Based Alignment for Machine Translation, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, pp. 80–87.

Giménez, J. and Márquez, L. (2007). Linguistic Features for Automatic Evaluation of Heterogenous MT Systems, *Proceedings of the Second Workshop on Statistical Machine Translation*, Association for Computational Linguistics, Prague, pp. 256–264.

Gimpel, K. and Smith, N. A. (2008). Rich Source-Side Context for Statistical Machine Translation, *Proceedings of the Third Workshop on Statistical Machine Translation*, Association for Computational Linguistics, Columbus, Ohio, pp. 9–17.

Hajič, J., Panevová, J., Hajičová, E., Sgall, P., Pajas, P., Štěpánek, J., Havelka, J., Mikulová, M., Žabokrtský, Z. and Ševčíková Razímová, M. (2006). Prague Dependency Treebank 2.0, LDC2006T01, ISBN: 1-58563-370-4.

Hutchins, J. (1954). The Georgetown-IBM Demonstration, *MT News International*, pp. 15–18.

Johnson, H., Martin, J., Foster, G. and Kuhn, R. (2007). Improving Translation Quality by Discarding Most of the Phrasetable, *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic, pp. 967–975.

Koehn, P. (2004). Statistical Significance Tests for Machine Translation Evaluation, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain, pp. 388–395.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A. and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation, Annual Meeting of the Association for Computational Linguistics. Demonstration session.

Kos, K. (2008). Adaptation of New Machine Translation Metrics for Czech, Bachelor's Thesis.

Kos, K. and Bojar, O. (2009). Evaluation of Machine Translation Metrics for Czech as the Target Language, *Prague Bulletin of Mathematical Linguistics* **92**: 135–147.

Leusch, G., Ueffing, N. and Ney, H. (2006). CDER: Efficient MT Evaluation Using Block Movements, *Proceedings of the European Association for Computational Linguistics*, pp. 241–248.

Lopez, A. (2007). Hierarchical Phrase-Based Translation with Suffix Arrays, *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Association for Computational Linguistics, pp. 976–985.

Manber, U. and Myers, G. (1990). Suffix Arrays: A New Method for On-Line String Searches, *The First Annual ACM-SIAM Symposium on Dicrete Algorithms*, pp. 319–327.

Mauser, A., Hasan, S. and Ney, H. (2009). Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models, *Proceedngs of the 2009 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 210–218.

Och, F. J. (2003). Minimum Error Rate Training in Statistical Machine Translation, *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 160–167.

Och, F. J. and Ney, H. (2002). Discriminative Training and Maximum Entropy Models for Statistical Machine Translation, *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 295–302.

Och, F. J. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models, *Computational Linguistics* **29**(1): 19–51.

Och, F. J. and Ney, H. (2004). The Alignment Template Approach to Statistical Machine Translation, *Computational Linguistics* **30**(4): 417–449.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation, *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, pp. 311–318.

Przybocki, M., Peterson, K. and Bronsart, S. (2008). Official results of the NIST 2008 "Metrics for MAchine TRanslation" Challenge (MetricsMATR08).

Sgall, P., Hajičová, E. and Panevová, J. (1986). *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*, Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands.

Shieber, S. M. and Schabes, Y. (1990). Synchronous tree-adjoining grammars, *Proceedings of COLING*, pp. 253–258.

Smith, A., Cohn, T. and Osborne, M. (2005). Logarithmic Opinion Pools for Conditional Random Ffields, *Proceedings of the Association for Computational Linguistics*, Ann Arbor, Michigan, pp. 18–25.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L. and Makhoul, J. (2006). A Study of Translation Edit Rate with Targeted Human Annotation, *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, The Association for Machine Translation in the Americas, Morristown, NJ, USA, pp. 223–231.

Stolcke, A. (2002). SRILM – An Extensible Language Modeling Toolkit, *Proceedings of International Conference on Spoken Language Processing*, pp. 901–904.

Su, K. and Wu, J. (1992). A New Quantitative Quality Measure for Machine Translation Systems, *Proceedings of the 14th International Conference on Computational Linguistics*, Nantes, France, pp. 433–439.

Sudoh, K., Watanabe, T., Suzuki, J., Tsukada, H. and Isozaki, H. (2008). NTT Statistical Machine Translation System for IWSLT 2008, *IWSLT 2008: Proceedings of the International Workshop on Spoken Language Translation*, pp. 92–97.

Tiedemann, J. (2009). News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces, *Proceedings of Recent Advances in NLP (RANLP 2009)*, pp. 237–248.

Tillmann, C., Vogel, S., Ney, H., Zubiaga, A. and Sawaf, H. (1997). Accelerated DP Based Search for Statistical Translation, *Proceedings of the 5th European Conference on Speech Communication and Technology*, Rhodes, Greece, pp. 2667–2670.

Turian, J. P., Shen, L. and Melamed, I. D. (2003). Evaluation of Machine Translation and its Evaluation, *Machine Translation Summit IX*, International Association for Machine Translation, pp. 386–393.

Yamada, K. and Knight, K. (2001). A Syntax-Based Statistical Translation Model, *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistic*, Toulouse, France, pp. 523–530.

Zaidan, O. (2009). Z-MERT: A Fully Configurable Open Source Tool for Minimum Error Rate Training of Machine Translation Systems, *The Prague Bulletin of Mathematical Linguistics* **91**: 79–88.

Žabokrtský, Z. and Bojar, O. (2008). TectoMT, developer's guide, *Technical Report TR-2008-39*, Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University in Prague.

Žabokrtský, Z., Ptáček, J. and Pajas, P. (2008). TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer, *Proceedings of the Third Workshop on Statistical Machine Translation*, Association for Computational Linguistics, Columbus, Ohio, pp. 167–170.

# Appendix A

# Semantic Part-of-Speech Types

| Semantic POS tag | Description |
|---|---|
| n.denot | denominating semantic noun |
| n.denot.neg | denominating semantic noun with which the negation is represented separately |
| n.pron.def.demon | definite pronominal semantic noun: demonstrative pronoun |
| n.pron.def.pers | definite pronominal semantic noun: personal pronoun |
| n.pron.indef | indefinite pronominal semantic noun |
| n.quant.def | definite quantificational semantic noun |
| adj.denot | denominating semantic adjective |
| adj.pron.def.demon | definite pronominal semantic adjective: demonstrative pronoun |
| adj.pron.indef | indefinite pronominal semantic adjective |
| adj.quant.def | definite quantificational semantic adjective |
| adj.quant.indef | indefinite quantificational semantic adjective |
| adj.quant.grad | gradable quantificational semantic adjective |
| adv.denot.ngrad.nneg | non-gradable denominating semantic adverb, impossible to negate |
| adv.denot.ngrad.neg | non-gradable denominating semantic adverb, possible to negate |
| adv.denot.grad.nneg | gradable denominating semantic adverb, impossible to negate |
| adv.denot.grad.neg | gradable denominating semantic adverb, possible to negate |
| adv.pron.def | definite pronominal semantic adverb |
| adv.pron.indef | indefinite pronominal semantic adverb |
| v | semantic verb |

Figure A.1: Semantic POS types.

# Appendix B

# User Documentation for zmert-moses.pl

The script *zmert-moses.pl* is used for launching the MERT algorithm to optimize the log-linear model parameters. It is used as interface which transforms user specified parameters into form suitable for Z-MERT,[1] an implementation of the MERT algorithm.

## B.1 Synopsis

The following parameters are recognized:

```
zmert-moses.pl input-text references decoder-executable decoder.ini
               --working-dir=STRING
               --rootdir=STRING
               --mertdir=STRING
               --jobs=N
               --mosesparallelcmd=STRING
               --old-sge
               --queue-flags=STRING
               --decoder-flags=STRING
               --filtercmd=STRING
               --no-filter-phrase-table
               --scorenbestcmd=STRING
               --inputtype=N
               --nbest=N
```

[1] http://www.cs.jhu.edu/~ozaidan/zmert/

```
--maxiter=N
--lambdas=STRING
--allow-unknown-lambdas
--lambdas-out=STRING
--activate-features=STRING
--predictable-seeds
--metric=STRING
--semposbleu-weights=STRING
--extract-sempos=STRING
--norm=STRING
--verbose
--mert-verbose=N
--decoder-verbose=N
--help
```

## B.2 Parameters

### Environment Settings

- `--working-dir=STRING`
  Name of the working directory where all files during the tuning phase are created. Default name is `mert-dir`.

- `--rootdir=STRING`
  Directory where Moses helper scripts reside. *zmert-moses.pl* is located in directory `rootdir/training`.

- `--mertdir=STRING`
  Directory with *zmert.jar*, which contains implementation of the MERT algorithm. Default location is `MOSES-DIR/zmert`.

- `--jobs=N`
  Run components capable of running in parallel with *N* jobs.

- `--mosesparallelcmd=STRING`
  Use a different script to run decoder instead of *moses-parallel.pl*.

- `--old-sge`
  Passed to moses-parallel, assume Sun Grid Engine $< 6.0$.

- `--queue-flags=STRING`
  Parameters that are passed to *qsub*, e.g. `'-l ws06osssmt=true'`. The default setting is `'-l mem_free=0.5G -hard'`. To reset the parameters, please use `--queue-flags=' '` (i.e. a space between the quotes).

- `--decoder-flags=STRING`
  Extra parameters for the decoder.

- `--filtercmd=STRING`
  Path to *filter-model-given-input.pl* script, which filters phrase tables to include only phrases in a given input text.

- `--no-filter-phrase-table`
  Disallow filtering of phrase tables. Useful if binary phrase tables are available.

- `--scorenbestcmd=STRING`
  Path to scoring script *score-nbest.py*.

## Z-MERT Settings

- `--inputtype=N`
  Handle different input types for Moses:

  - 0 - text,
  - 1 - confusion network,
  - 2 - lattices (not implemented yet).

  Default value is 0.

- `--nbest=N`
  Size of n-best list that is used by MERT. Default value is 100.

- `--maxiter=N`
  Maximum number of MERT iterations. The number of iterations is unlimited by default.

- `--lambdas=STRING`
  Default values and ranges for lambdas. A complex string like `'d:1,0.5-1.5 lm:1,0.5-1.5 tm:0.3,0.2-0.7;0.2,0.2-0.7; w:0,-0.5-0.5'` is expected.

- `--allow-unknown-lambdas`
  Keep going even if someone supplies a new lambda in the `--lambdas` option (such as `'newmodel:1,0-1'`). Optimize the lambda with starting and minimum/maximum value as defined in `--lambdas`.

- `--lambdas-out=STRING`
  File where final lambdas should be written.

- `--activate-features=STRING`
  Comma-separated list of features to optimize. Optimize all features if not defined.

- `--predictable-seeds=N`
  Provide predictable seeds to MERT so that random restarts are the same on every run.

- `--metric=STRING`
  Metric name used for optimization including metric parameters such as `'BLEU 4 closest'` or `'SemPOS 0 1'`. Use default metric parameters by specifying only the metric name, e.g. `BLEU` or `SemPOS`. Supported metrics are

  – BLEU,
  – TER,
  – TER-BLEU (TER minus BLEU),
  – SemPOS,
  – SemPOS_BLEU (SemPOS plus BLEU).

  SemPOS metric requires 2 parameters:

  1. factor position of `t-lemma`,
  2. factor position of `sempos` in input text.

  Default values are `'0 1'`.

  SemPOS_BLEU metric requires 7 parameters:

  1. weight of SemPOS,
  2. weight of BLEU,
  3. factor position of `lemma` for SemPOS in input,
  4. factor position of `sempos` for SemPOS in input,

5. max n-gram for BLEU,
6. reference length strategy for BLEU,
7. input position of factor to compute BLEU.

Default parameters are `'1 1 1 2 4 closest 0'`. For more details about default BLEU and TER parameters see Z-MERT documentation.[2]

- `--semposbleu-weights=STRING`
  Weights for SemPOS and BLEU in format `N:M` where `N` is SemPOS weight and `M` BLEU weight. This parameter has effect only if `--metric=SemPOS_BLEU`.

- `--extract-sempos=STRING`
  Specify how factors required by the evaluation metric are extracted from the decoder output:

  - `none` - decoder generates all required factors,
  - `factors:<factor_list>` - extract factors with index in <factor_list> from decoder output, e.g. `factors:0,2,3` to extract the first, third and fourth factor from the decoder output. If this option is used the factor extraction precedes the default parameter extraction specified in the `--metric` parameter.
  - `tmt` - use TectoMT[3] to generate required factors from surface word forms.

  This parameter should be used only with SemPOS or SemPOS_BLEU metric.

- `--norm=STRING`
  Select lambdas normalization for Z-MERT which is applied after each iteration:

  - `none` - no normalization,
  - `absval 1 lm` - scale weights so that the weight for lm equals 1,
  - `maxabsval 1` - scale weights so the maximum absolute value is 1,
  - `minabsval 1` - scale weights so the minimum absolute value is 1,
  - `LNorm 2 1` - scale weights so that the L-2 norm equals 1,

  In the above, lm can be replaced by any other feature name, and the numbers can also be replaced by other numerical values.

[2] http://www.cs.jhu.edu/~ozaidan/zmert/
[3] http://ufal.mff.cuni.cz/tectomt

## Verbosity

- `--verbose`
  Turn on verbose mode in zmert-moses.pl.
- `--mert-verbose=N`
  Z-MERT verbosity:
  - `0` - only decoder warnings and error messages are included,
  - `1` - default level of verbosity,
  - `2` - even more verbose output.
- `--decoder-verbose=N`
  Decoder verbosity:
  - `0` - decoder output is ignored,
  - `1` - decoder output (STDOUT and STDERR) is integrated into own output.
- `--help`
  Print help for zmert-moses.pl.