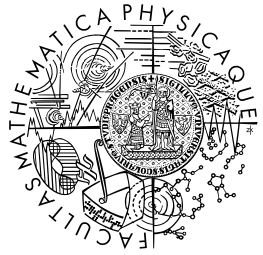


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Mgr. Aleš Tamchyna

Lexical and Morphological Choices in Machine Translation

Institute of Formal and Applied Linguistics

Supervisor: RNDr. Ondřej Bojar, Ph.D.

Study Program: Computer Science
Specialization: Mathematical Linguistics

Prague 2017

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

Prague, 12. 4. 2017 Aleš Tamchyna

Title:	Lexical and Morphological Choices in Machine Translation
Author:	Mgr. Aleš Tamchyna
Department:	Institute of Formal and Applied Linguistics
Supervisor:	RNDr. Ondřej Bojar, Ph.D.
Keywords:	machine translation, discriminative models, machine learning, morphology

Abstract: This work focuses on two problems in machine translation: lexical choice and target-side morphology. The first problem is the correct transfer of meaning from the source language to the target language. The second problem, which is mainly relevant for morphologically rich target languages, is then the choice of the correct surface form of each target lexeme. We work with these problems within the framework of phrase-based machine translation and we propose a discriminative model of translation which utilizes both source and target context information and which uses rich linguistically motivated features. We show how our model addresses specific weaknesses of standard phrase-based systems and that it provides consistent improvements of translation quality across a broad range of experiments. Apart from our main contribution, we also provide a number of experimental evaluations, analyses and manual annotation experiments, mostly related to English-Czech translation.

Název práce:	Lexikální a tvaroslovné varianty ve strojovém překladu
Autor:	Mgr. Aleš Tamchyna
Ústav:	Ústav formální a aplikované lingvistiky
Vedoucí práce:	RNDr. Ondřej Bojar, Ph.D.
Klíčová slova:	strojový překlad, diskriminativní modely, strojové učení, morfologie

Abstrakt: Práce se zabývá dvěma problémy strojového překladu: lexikální volbou a morfologií v cílovém jazyce. První úlohou je správné přenesení významu ze zdrojového jazyka do cílového. Druhá úloha, která hraje roli především při překladu do tvaroslovně bohatých jazyků, je pak správná volba povrchové formy u cílových lexémů. Tyto úlohy řešíme v rámci frázového strojového překladu. Navrhujeme diskriminativní překladový model, který využívá lingvisticky motivované rysy extrahované jak ze zdrojového, tak z cílového kontextu. Ukazujeme, že tento model řeší konkrétní slabiny standardních frázových systémů. Pomocí řady experimentů pak dokládáme, že model konzistentně zlepšuje kvalitu výsledného překladu. Vedle tohoto hlavního příspěvku popisujeme analýzy, ruční anotace a experimenty zaměřené především na anglicko-český překlad.

Contents

List of Figures

List of Tables

Acknowledgement

I would like to thank my advisor, RNDr. Ondřej Bojar, Ph.D., for the countless hours that he dedicated to helping me with my research. His invaluable advice and many thought-provoking questions have made this work possible.

I am grateful to my colleagues at the Institute of Formal and Applied Linguistics who have helped me and inspired me throughout my doctoral studies. I was amazed time and again by their readiness to discuss my research problems and to offer insightful advice which often helped me find a way forward.

I would like to thank Dr. Alexander Fraser for the opportunity to visit Ludwig Maximilian University of Munich, and for his many insights which helped complete this research.

I am grateful to my father for his endless patience and support.

Finally, I would like to thank my girlfriend Veronika who has always stood by my side.

This work was supported by the grants H2020-ICT-2014-1-645452 (QT21), H2020-ICT-2014-1-644402 (HimL), H2020-ICT-2014-1-644753 (KConnect), FP7-ICT-2011-7-288487 (MosesCore) and FP7-ICT-2010-6-257528 (Khresmoi) of the European Union, by SVV projects 260104, 267314, 260224, and by project 1356213 of Charles University Grant Agency.

1 Introduction

Machine translation (MT) is an important task in natural language processing (NLP). This work focuses on two problems in MT: lexical choice and target-side morphology.

The first problem is the correct transfer of meaning from the source language to the target: when translating a foreign word, the system should disambiguate its sense in the source language and choose a *lexeme* in the target language which best corresponds to its meaning.

The second problem is then the choice of the correct *surface form* of each lexeme. This task is mainly relevant for target languages with rich morphology where multiple surface forms can correspond to a single lexeme. Surface forms may carry information about gender, grammatical number, case and other morpho-syntactic attributes. The correct choice then depends both on the source sentence (e.g. transfer of singular/plural for nouns) and on the target sentence (e.g. the grammatical case of nouns may depend on the valency frame of the target verb).

We work with these problems within the framework of phrase-based machine translation (PBMT), the predominant approach to MT. At the time of writing, PBMT is gradually being replaced in the research community by deep learning. However, lexical choice and target-side morphology represent the basic challenge of MT, and we therefore believe that many of the findings in this work are more general and can be relevant even for the newly emerging approaches to MT.

1.1 Motivation

We address the two tasks (lexical and morphological choice) with a single discriminative model which operates on the level of phrases and which uses context information from both the source and target sentence. In this work, we will refer to the proposed model as DPTM (discriminative phrasal translation model).

While we use a single model jointly for both tasks, it is useful to motivate the use of context information separately for lexical choice and for target-side morphology.

Lexical choice can be viewed as a specific setting of *word sense disambiguation* (Vickrey et al., 2005; Carpuat and Wu, 2007): the various possible translations of a word (or a phrase) in the target language can be thought of as the senses of the source-side word/phrase to be disambiguated. For instance, we can identify senses of the English word "shoot" by their Czech translations "střelba" (shooting a weapon) or "natáčení" (shooting a film). It is well known in the field of computational linguistics that context information is essential for disambiguating the word sense. We include source-context information in DPTM exactly for this purpose.

Morphological choice requires additional sources of information. When selecting the correct surface form of a word, the system needs some information from the source sentence (for instance, to select the grammatical number – singular or plural) but also from the target sentence which is currently being constructed.

PBMT is not capable of correctly modelling lexical choice or morphological coherence by itself. We provide specific examples in Section 3.3 which illustrate this issue and clearly motivate the use of context information beyond the scope of what standard PBMT systems can capture.

The model that we propose explicitly takes context information into account and attempts to avoid the problems of standard PBMT systems. We describe how DPTM can handle the motivating examples in Chapter 5.

1.2 Contributions

The main contribution of this work is the implementation and evaluation of a discriminative model of translation which utilizes both source and target context information and which uses rich linguistically motivated features. We show how our model addresses specific weaknesses of standard PBMT sys-

tems and that it provides consistent improvements of translation quality across a broad range of experiments.

Apart from this main contribution, we also describe and analyze the English-Czech system Chimera (Bojar et al., 2013b) which we helped develop. We provide a number of experimental evaluations, analyses and manual annotation experiments. However, it should be noted that the design of Chimera itself is *not* our contribution.

We also briefly document several research ideas which were unsuccessful but which may still prove interesting for some readers.

1.3 Background of the Work

The basic setting of this work was proposed and partially explored during the JHU Summer Workshop in 2012. The team topic was Domain Adaptation in Machine Translation. In this joint effort, we developed the first integration of a discriminative model in the Moses toolkit (Koehn et al., 2007).[1] The model used only source context information. In parallel, the integration was also carried out for hierarchical phrase-based translation which brings other technical challenges.

During the workshop, the team never managed to obtain positive results due to a number of bugs and design problems in the implementation. The work and its state at the end of the workshop is described in the final report (Carpuat et al., 2012). We also described the original implementation along with preliminary results in a follow-up article (Tamchyna et al., 2014a). The experiments presented in that work were carried out by Fabienne Braune and Alexander Fraser.

Some members of the original team continued the work on discriminative models further. For hierarchical PBMT, the effort of Fabienne Braune eventually led to the publication of positive results (Braune et al., 2016). Our contribution to the work on hierarchical MT is mainly the common codebase (shared between phrase-based and hierarchical approaches) which was developed at the workshop.

Our main focus has been the phrase-level discriminative model. Together with Marcin Junczys-Dowmunt, we re-implemented the integration of the phrasal discriminative model in Moses. The refactored code was then included in the main branch of Moses.

[1] Moses is perhaps the most widely-used implementation of PBMT with a developer community that includes prominent researchers in the field.

We then continued experimenting with and extending this model. Our most significant extension is the addition of target-side context information to the model, which requires some additional implementation effort for efficiency. This extension is described in a joint conference paper (Tamchyna et al., 2016a).

1.4 Outline

We introduce the basic concepts of phrase-based MT in Chapter 2. In Chapter 3, we describe the Chimera system for English-Czech translation. Until recently, Chimera could be considered state-of-the-art for English-Czech MT and it forms our baseline. We also analyze its properties and limitations in this chapter, providing motivation for our approach. We describe the machine learning methods relevant for our work in Chapter 4.

Chapter 5 introduces our discriminative model (DPTM). Its full integration into a PBMT system proved to be a technical challenge which we describe in Chapter 6. The main experiments are presented in Chapter 7.

We discuss related work in Chapter 8. Chapter 9 describes possible future extensions and applications of DPTM and concludes the thesis.

2

Phrase-Based Machine Translation

Phrase-based MT (PBMT, Koehn et al. 2003) has been, until recently, the most popular approach to MT. We focus PBMT in this work and this chapter describes some of its basic concepts. Topics which are directly relevant for this work are described in more detail. Koehn (2010) is an excellent source of further information on phrase-based MT.

The essential ingredient for building a phrase-based translation system are parallel data: a collection of sentences in the source language and of their translations in the target language.

We denote our collection of target sentences $E = (\boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(N)})$ and the source sentences $F = (\boldsymbol{f}^{(1)}, \ldots, \boldsymbol{f}^{(N)})$. We use the letters e and f as is customary in MT literature (e originally stood for English and f for French, or *foreign*). Each sentence consists of words: $\boldsymbol{e}^{(i)} = (e_1^{(i)}, \ldots, e_n^{(i)})$ and $\boldsymbol{f}^{(i)} = (f_1^{(i)}, \ldots, f_m^{(i)})$.

Parallel training data is invaluable because it allows the statistical MT system to learn the correspondences between source-language units (words, phrases, constituents,...) and target-language units. In the next section, we will sketch how correspondences between words may be obtained from this data.

2.1 Word Alignment

Word alignment is an essential first step in the training of phrase-based translation models. The techniques used for word alignment today were originally developed for statistical *word-based* translation – in particular, word alignment uses the so-called IBM models (Brown et al., 1993). The mathematical background of these models is beyond the scope of this work.

Here we only note that the IBM models attempt to capture translation in a probabilistic framework where the mapping between words is a latent variable. The optimization procedure attempts to find such an alignment that best explains the training data, i.e. that makes the parallel corpus as probable as possible.

In today's practice, this optimization is done in both directions (aligning English words to *foreign* words and vice versa) and the two mappings are then *symmetrized* using one of several possible heuristics.

After the word alignment step, we have the parallel corpus and a (possibly) many-to-many mapping between source-side and target-side words in each sentence.

2.2 Phrase Extraction

Once the word alignment of the parallel data is available, phrase pairs are extracted from the training corpus. In phrase-based MT, a phrase is simply a continuous sequence of words. Phrases do not necessarily correspond to any meaningful linguistic units.

We describe this step in more detail because it is highly relevant for the feature extraction procedure of DPTM. Note that there are many possible methods for obtaining phrase translation tables from parallel data. The one that we discuss here is the most common approach.

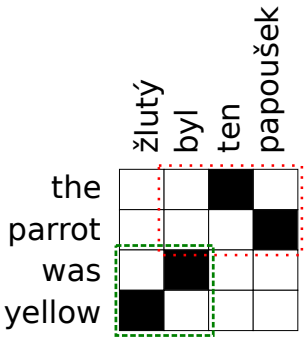


Figure 2.1: Extraction of phrase pairs from an English-Czech sentence pair "the parrot was yellow", "žlutý byl ten papoušek".

The phrase extraction algorithm goes over every sentence in the parallel data and obtains *all* admissible phrase pairs from it. A phrase pair is admissible when it is *consistent* with the word alignment.

The consistency criterion is a heuristic which is best illustrated by an intuitive example. In Figure 2.1, we show a short English-Czech sentence pair and

the word alignment represented as a matrix of Boolean values. A phrase pair is consistent with the alignment if it contains at least one aligned word pair and if it fulfills the following condition: all words in the source phrase are aligned only to words inside the target phrase, and vice versa.

In the example, the phrase pair "was yellow", "žlutý byl" is consistent: none of the English words are aligned outside the Czech phrase and no Czech words are aligned outside the English phrase. On the other hand, the phrase pair "the parrot", "byl ten papoušek" is *not* consistent because the Czech word "byl" is aligned to "was" which is not part of the English phrase.

Note that some words can be unaligned. There are slight differences among implementations of phrase extraction with regard to unaligned words – some allow phrase pairs with unaligned words at phrase boundaries to be extracted while others do not.

2.3 Phrase Table

The phrase table is a probabilistic phrasal dictionary which is at the core of a PBMT system.

The phrase extraction algorithm simply outputs a list of all phrase pairs found in the training data. To construct a phrase table from this list, phrase pair occurrences are counted and phrasal translation probabilities are estimated given the counts as follows:

$$P(\bar{e}|\bar{f}) = \frac{c(\bar{e} \wedge \bar{f})}{c(\bar{f})} \tag{2.1}$$

$$P(\bar{f}|\bar{e}) = \frac{c(\bar{e} \wedge \bar{f})}{c(\bar{e})} \tag{2.2}$$

Here we use $\bar{f}$ and $\bar{e}$ to denote the source and target phrase, respectively. The function c represents the *counts* of phrase (co-)occurrences in the training data.

The final phrase table usually contains the phrase pairs, their direct and inverse phrasal translation probabilities, direct and inverse lexical weights (a smoothed variant of phrase translation probabilities based on lexical translation probabilities), word alignment information and possibly other data, such as raw phrase counts (from which the probabilities were estimated).

2.4 Language Model

The language model (LM) is another essential component of a phrase-based system. We first introduce the noisy channel formulation of the translation problem from which the need for a LM follows directly.

The noisy channel model (Shannon, 2001) can be used to connect MT and information theory; we can view translation as communication over an imperfect communication channel. The input sentence $\boldsymbol{f}$ is considered to be originally sent in English as sentence $\boldsymbol{e}$. During transmission, noise is added and $\boldsymbol{e}$ arrives in a foreign language.

The task of translation is then to *reconstruct the original sentence*, i.e. we look for the most probable sentence $\boldsymbol{e}$ given the input $\boldsymbol{f}$:

$$\begin{aligned}\hat{\boldsymbol{e}} &= \arg\max_{\boldsymbol{e}} P(\boldsymbol{e}|\boldsymbol{f}) \\ &= \arg\max_{\boldsymbol{e}} \frac{P(\boldsymbol{f}|\boldsymbol{e})P(\boldsymbol{e})}{P(\boldsymbol{f})} \\ &= \arg\max_{\boldsymbol{e}} \underbrace{P(\boldsymbol{f}|\boldsymbol{e})}_{TM}\underbrace{P(\boldsymbol{e})}_{LM}\end{aligned} \tag{2.3}$$

The first equality holds because of Bayes' theorem. Since we are only interested in the most probable result (argmax) and $P(\boldsymbol{f})$ is independent of $\boldsymbol{e}$, we can disregard the denominator.

The two probability distributions correspond to the noisy channel view: the source generates a sentence $\boldsymbol{e}$ with a probability $P(\boldsymbol{e})$. This sentence is then sent over the noisy channel which transforms $\boldsymbol{e}$ into $\boldsymbol{f}$ with the probability $P(\boldsymbol{f}|\boldsymbol{e})$. This decomposition gives us the two basic components: the *translation model* (TM) and the *language model*.

The task of language modelling is to estimate the probabilities of strings in the target language. We describe LMs in some detail here because the model that we propose in this work is closely related.

We can use the chain rule of probability to arrive at a more convenient decomposition:

$$P(\mathbf{e}) = P(e_1) \cdot P(e_2|e_1) \cdot P(e_3|e_1, e_2) \cdots P(e_l|e_1, \ldots, e_{l-1}) \tag{2.4}$$

In this formula, we predict one word at a time and we condition the prediction on previous words in the sequence. State-of-the-art LMs are currently

based on deep neural networks (recurrent and/or convolutional, Józefowicz et al., 2016) which indeed utilize the whole history of words.

However, in phrase-based MT, the common approach to language modelling are still count-based n-gram models which we further describe here. These models make an additional strong independence assumption:

$$P(\mathbf{e}) \approx P(e_1) \cdot P(e_2|e_1) \cdots P(e_l|e_{l-n}, \ldots, e_{l-1}) \tag{2.5}$$

The equation states that we can approximate the true conditional probabilities by only looking back at a *fixed-sized* window of n words (this is the so-called Markov assumption). Typically, n is around 2-4 words. When a model looks at a history of 2 words, we call it a trigram (3-gram) LM because the full size of the n-gram is three words.

Note that Equation 2.5 mixes n-grams of different lengths: we begin with an unconditional unigram probability $P(e_1)$, continue with a bigram $P(e_2|e_1)$ etc. This is typically not done in practice. Instead, in order to keep the n-gram length constant and also to explicitly account for sentence beginning and end, special tokens "<s>" and "</s>" are usually added to the sequence. For example, the probability of the sentence "thank you very much" according to a trigram LM would be the following:

$$\begin{aligned} P(\text{"thank you very much"}) = {} & P(\text{"thank"}|\text{"<s><s>"}) \\ & \times P(\text{"you"}|\text{"<s>thank"}) \\ & \times P(\text{"very"}|\text{"thank you"}) \\ & \times P(\text{"much"}|\text{"you very"}) \\ & \times P(\text{"</ s>"}|\text{"very much"}) \end{aligned}$$

The maximum likelihood estimate of n-gram probabilities is again based on counts in the corpus:

$$P(e_i|e_{i-n}, \ldots, e_{i-1}) = \frac{c(e_{i-n}, \ldots, e_i)}{c(e_{i-n}, \ldots, e_{i-1})} \tag{2.6}$$

For instance, to estimate $P(\text{"very"}|\text{"thank you"})$, we simply count the number of occurrences of the sequence "thank you very" in the training data. We then divide this count by the number of occurrences of the prefix "thank you" (followed by anything).

Observations of long word sequences are sparse and smoothing is required. The most common method for n-gram LMs is modified Kneser-Ney smoothing (Chen and Goodman, 1999).

2.5 Log-Linear Model

The noisy channel formulation can be generalized. We lose the information-theoretic interpretation of the model but obtain a much more flexible framework.

Going back to Equation 2.3, we can decide to give one of the probability distributions a different weight by raising its value to the power of λ. In log space, this is equivalent to multiplying the log-probability.

We can also add other features which could help discriminate good and bad translation candidates. We end up with a product of many scores f_i, each with a parameter λ_i which controls its relative importance. When we move to log space, our model score becomes a weighted sum of feature values. We arrive at the formulation of the log-linear model:

$$\hat{\boldsymbol{e}} = \arg\max_{\boldsymbol{e}} \exp \sum_i \lambda_i f_i(\boldsymbol{e}, \boldsymbol{f}) \tag{2.7}$$

Note that in the special case where we only have two features (TM and LM), represent them as log-probabilities and set both λ_i to one, this formulation is equivalent to the noisy channel model. In practice, we omit the final exponential because it is a monotonic function (and therefore it cannot change the ranking of translation candidates).

Phrase-based translation makes several assumptions and approximations which we will informally state here because they affect the probabilistic interpretation of the model score and of individual features. Our discussion loosely follows the introduction to PBMT in Zens (2008).

The argmax in Equation 2.7 is defined over English sentences $\boldsymbol{e}$. In PBMT, we decompose the input sentence into phrases. Each English sentence that our model can produce is in fact the result of the following decisions:

1. How to segment $\boldsymbol{f}$ into phrases $\bar{f}_i$,
2. how to order these phrases in the translation $\boldsymbol{e}$,
3. how to translate the phrases $\bar{f}_i$ into the target phrases $\bar{e}_i$.

A single possible translation $\boldsymbol{e}$ can often be constructed from $\boldsymbol{f}$ in different ways by selecting a different phrasal segmentation and reordering. In theory,

the phrase-based system should therefore sum over all possible phrasal segmentations and orderings. This is not done in practice. Instead, PBMT approximates the sum by the maximum: the most probable segmentation.

Without providing a rigorous definition of segmentation and reordering, let us denote both as a single latent variable S. Our discussion so far can be formulated as follows:

$$\hat{\boldsymbol{e}} = \arg\max_{\boldsymbol{e}} \sum_i \lambda_i f_i(\boldsymbol{e}, \boldsymbol{f}) \tag{2.8}$$

$$= \arg\max_{\boldsymbol{e}} \sum_S \sum_i \lambda_i f_i(\boldsymbol{e}, S, \boldsymbol{f}) \tag{2.9}$$

$$\approx \arg\max_{\boldsymbol{e}} \max_S \sum_i \lambda_i f_i(\boldsymbol{e}, S, \boldsymbol{f}) \tag{2.10}$$

We begin with the log-linear model formulation. Equation 2.9 makes the marginalization over the values of the latent variable S explicit. Equation 2.10 then makes the maximum approximation to the sum.

This equation formalizes the fact that feature scores depend not only on the actual translation but also on the segmentation and reordering. Indeed, the total phrasal translation probability $P_{TM}(\boldsymbol{e}, S|\boldsymbol{f}) \propto \prod_{(\bar{e}_i, \bar{f}_i)} P_{TM}(\bar{e}_i|\bar{f}_i)$ is different for different segmentations of $\boldsymbol{f}$. Similarly, the distortion model score (see below) clearly depends on phrasal reordering.

In our work, we essentially develop a new component of the log-linear model. The distinction that we discuss here is important in our case because it provides a mathematical description of what our model predicts and what is given.

Features

In this section, we briefly describe the features commonly used in the log-linear model in PBMT.

In a typical setting, the model includes the four phrase table scores (see Section 2.3), LM probability and several other scores which we introduce now:

- **Phrase penalty** is a function which simply adds 1.0 for each phrase that the system produces. This allows the system to regulate the average length of phrases used: if the penalty is very low (or even negative), the system is encouraged to produce many short phrases, and vice versa.

- **Word penalty** is a similar feature. Its general purpose is to regulate the average length of translations – some languages use more words on average than others. By explicitly counting the words, we allow the model to directly control the translation length.

- **Distortion penalty** is a basic form of a reordering model. It penalizes (or encourages, when set to a small or negative value) when the system "jumps around" the input sentence. It is usually calculated as the (absolute value of) the difference between the end position of the previous input phrase and the beginning of the current input phrase.

Weight Optimization

So far we have introduced the mathematical formulation of the log-linear model and briefly described the features that it typically includes in PBMT. Each feature f_i has an associated weight λ_i and in this section, we briefly discuss how the weight values are optimized.

The weights are typically set to maximize translation quality on a held-out set of parallel sentences. In order to address this task, we need a way to automatically calculate the quality of a translation and an algorithm to search for a good set of weights.

There are many possible automatic metrics of translation quality. Most require *reference* translations of the held-out set (usually produced by professional translators) against which MT outputs are compared by measuring some sort of overlap, see e.g. Bojar et al. (2016c). The most common metric is BLEU (Papineni et al., 2002), even though its limitations are well known.

Similarly, there are many optimization algorithms for finding the model weights. Note that this optimization problem is difficult: the error function (e.g. BLEU) is piece-wise constant, preventing the direct use of gradient-based optimizers. Moreover, the evaluation of model parameters may require re-translating the held-out set, which is computationally expensive.

The most popular approach to weight optimization (or *tuning*) has traditionally been minimum error rate training (MERT, Och 2003). Successful alternatives have been proposed (Hopkins and May 2011; Cherry and Foster 2012, *inter alia*) but we opt for using MERT in our work.

2.6 Decoding

We have briefly introduced the underlying models of PBMT. In this section, we turn our attention to the problem of decoding, i.e., finding the most probable translation of an input sentence under the phrase-based model. We describe decoding in more detail because our model is tightly integrated in the phrase-based search. When implementing the integration, we encounter several technical challenges which can only be described given sufficient background.

Translation Options

The first step of decoding goes through all possible segmentations of the source sentence (here, we mean by segmentation the division of the sentence into continuous spans). For each source-side span, the phrase table is consulted to obtain all possible translations of the span.

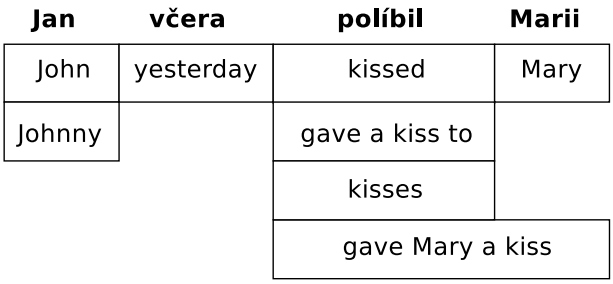


Figure 2.2: Example of translation options for a single Czech sentence.

Figure 2.2 shows an example of the result. Candidate translations are found for various overlapping source spans. Once the set of *translation options* is available, the task is to find a selection of phrasal translations and determine their order such that:

- each source word is covered exactly once, and
- the model score of the translation is as high as possible.

Beam Search

We now describe the search for the best translation hypothesis. This search is not exact; pruning is required for computational tractability. Similarly to other parts of the phrase-based pipeline, there are several possible solutions of the search problem. Here we describe the "standard" stack-based beam search algorithm.

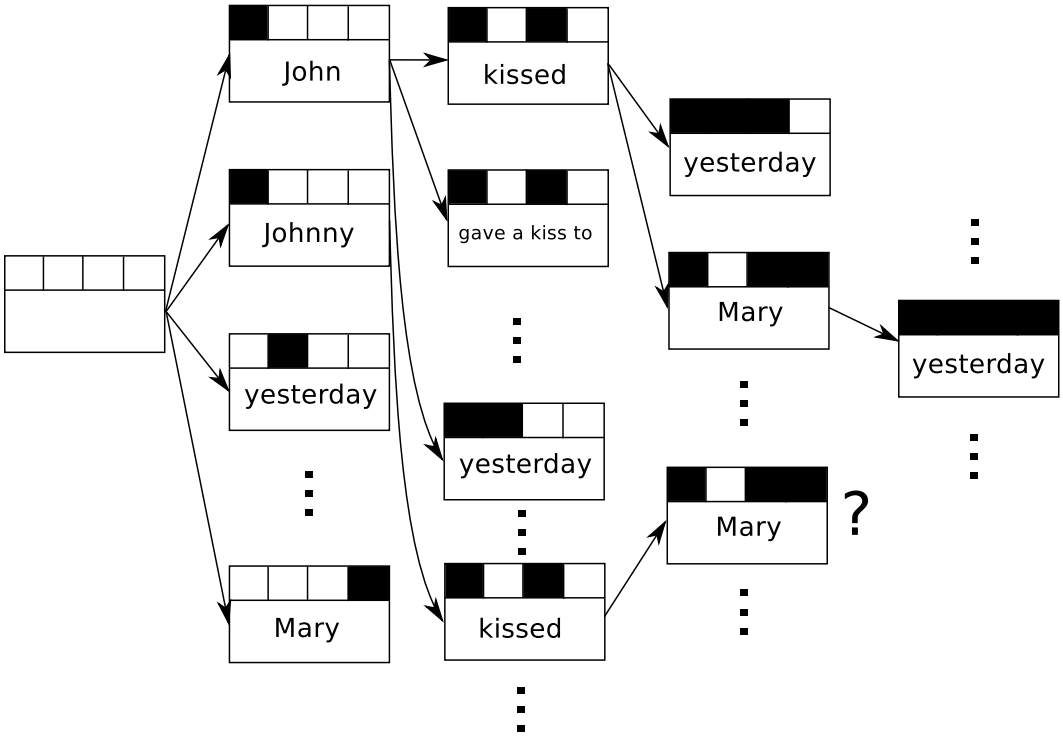


Figure 2.3: Decoding in phrase-based MT.

Figure 2.3 gives an example of a (subgraph of) search graph that the decoder explores. The input sentence and the translation options are the same as in Figure 2.2.

The left-most state is the starting point: an empty hypothesis. No output has been produced so far and no part of the input has been covered. The empty squares illustrate the Boolean *coverage vector* that keeps track of which input words have already been translated.

The first hypothesis expansion[1] translates a single input phrase. All possible starting positions in the source sentence are considered and evaluated. The translation is produced left to right: the first translated phrase will be the first phrase in the output.

As decoding progresses, we gradually expand the partial hypotheses to cover more and more of the input sentence until we obtain the best translation candidate.

We need to prune the partial hypotheses during search based on their model score (we discard the least promising translations). The overall hypothesis

[1]By the term *hypothesis expansion*, we refer to extending the current *partial hypothesis* by translating an uncovered source phrase.

score decreases as it covers more of the input sentence. Partial hypotheses are therefore organized in *stacks* depending on how many source words they have already covered (i.e., the sum of the coverage vector) which prevents short partial translations from competing with longer outputs (which naturally have a lower probability).

The phrase-based search explores the graph, organizing partial hypotheses on stacks and pruning. Once all the surviving translations have been evaluated, the final stack contains the top-scoring possible translations of the full sentence. The subgraph in Figure 2.3 shows only a single full translation.

Scoring of Partial Translations

The step of scoring of partial translation hypotheses is crucial for our work. Recall that we introduced various *features* in the log-linear model (Section 2.5) which together form the model score (the score is their weighted combination). So far we have always formulated the problem as scoring the full pair of sentences $(\boldsymbol{e}, \boldsymbol{f})$, however in reality, we need to evaluate these features for partial outputs.

This introduces the problem of decomposing the features over decoder decisions. For example, the standard phrasal translation probability $P_{TM}(\boldsymbol{e}|\boldsymbol{f})$ decomposes very naturally: each phrase pair has a fixed probability estimated from the training data which is independent of the rest of the translation. The feature can therefore be evaluated for phrase pairs in isolation (locally). Such features are called *stateless*. Word penalty or phrase penalty are also examples of stateless features.

The LM, on the other hand, is a *stateful* feature. In order to determine the probability of a hypothesis expansion, the LM needs to see the previously generated words.[2] This is intuitively obvious: for example, the probability of producing a comma should be much lower when we are at the beginning of a sentence as opposed to somewhere in the middle.

By *state*, we mean the information from the previous output required for the evaluation of the feature. In the case of a trigram LM, a hypothesis state would be the last two words of the partial translation. No words before that can affect the final LM score any more.

Note that unlike the TM, the LM has to evaluate the same phrasal translation multiple times during the search. Consider the word (phrase) "kissed" in Figure 2.3 – two LM evaluations of the phrase are shown in the subgraph, one

[2]Recall that the probability of a sentence according to a LM is a product of conditional probabilities, each predicting a single word based on its (limited) history.

for the context "<s> John" and the other for "<s> Johnny". In practice, the LM is evaluated many times for each target phrase as phrases typically appear in many different contexts during the search.

This implies that LM evaluation is much more computationally expensive than TM evaluation (which is in fact done even before decoding, when translation options are loaded from the phrase table). This is evidenced by the amount of implementation effort to make LMs efficient (see e.g. Heafield 2011). Even with the many optimizations currently included in state-of-the-art LM toolkits, LM evaluation still accounts for a large part of decoding time in PBMT systems.

Recombination and Pruning

The notion of a state is important for decoding because it informs the decoder when two partial translations can be considered equivalent. Note the question mark in Figure 2.3: the partial translation "Johnny kissed Mary" is very similar to "John kissed Mary". At this point in the search, no feature will be able to distinguish between them (assuming again only a trigram LM).

The decoder tests partial translations for equality (in this sense) by comparing their coverage vectors and feature states. If they are all identical, the hypotheses are *recombined* and they are treated as a single hypothesis from this point onward. Hypothesis recombination makes decoding much more efficient but still does not eliminate the need for pruning.

Pruning occurs multiple times during translation. Even when possible translations are loaded from the phrase table, a limit of maximum translations per source phrase is usually applied which discards phrase pairs with low model score according to the weighted combination of *stateless* feature scores.

However, the most important stage of pruning occurs during search. Each stack has a limited size. When a new partial hypothesis is added to the stack, its state is first compared with existing hypotheses so that it can potentially be recombined. If it does not match with any hypothesis, the decoder attempts to add it to the stack which may already be full. Pruning simply discards the lowest scoring hypothesis in the stack in this situation. (Note that different pruning strategies can be employed, notably *threshold pruning* which requires hypothesis not to be worse than the current best hypothesis by a certain ratio.)

Future Cost

Consider two competing partial translations which have covered the same *number* of input words (and therefore share a stack) but they differ in which *part*

of the source sentence they have translated. One partial translation may have translated only the easy parts so far; there is less entropy in its decisions and it therefore has a higher probability than the other hypotheses, which already made some difficult decisions (i.e. decisions with a lower overall probability).

Future cost is a method of preventing these "cheaper" translations from filling the stacks and thereby discarding all the "difficult" translations. Because even bad translations of easier spans can have a higher probability than good translations of difficult parts of the input, pruning would introduce search errors.

The future cost is added to the model score of each partial hypothesis. It is a heuristic estimate of the cheapest possible cost of finishing the translation from the given partial hypothesis. The "cheap" partial hypotheses obtain a higher future cost because of the low-probability translations they will have to produce in the future.

2.7 Factored Translation

In this section, we briefly introduce factored phrase-based MT (Koehn and Hoang, 2007) which we utilize in our work. Factored MT was introduced in as a way to make PBMT more linguistically plausible. Words are not represented simply as tokens but instead as vectors of *factors*. Each word can therefore include not only its surface form but also the lemma, morphological tag or any other type of information.

Factored MT enables complex scenarios which involve a sequence of translation and generation steps. See Tamchyna (2012) for a thorough description of this paradigm and the evaluation of its various possible configurations.

In this work, we avoid complex factored setups and only use factored MT as a way of including additional information about words both on the source side and on the target side. In the source, the information is mainly used for generating features used by DPTM. On the target side, we also employ the factors for language modelling in some experiments.

3

Empirical Analysis of English-Czech Phrase-Based Translation

We describe and analyze Chimera (Bojar et al., 2013b; Tamchyna et al., 2014b; Bojar and Tamchyna, 2015; Tamchyna et al., 2016b), a hybrid translation system built around a phrase-based component. It was developed as a submission to the WMT news translation shared task. So far, we participated in years 2013-2016 and achieved first place every time until 2016 (Bojar et al., 2013a, 2014, 2015, 2016a). Considering these results, we could say that until recently, Chimera represented the state of the art for English-Czech translation.

We therefore see Chimera as our ultimate baseline. Note however that because it is a hybrid system, improving over its performance by adding discriminative models would be a strong result.

While DPTM should ideally achieve this goal, a more modest aim for our work is to move some of the advantages of the hybrid design directly into the statistical (phrase-based) component – by not relying on external tools and the rather complex system combination, the translation system would become much more practical. And perhaps more importantly, some parts of the combination are language-dependent; by "simulating" their effect with DPTM, we could make the approach more generally applicable in terms of language pairs.

We contributed to the system-building effort when developing Chimera; the original idea for the combination is not ours.

3.1 Design of Chimera

Chimera is a combination of three major components:

- a factored phrase-based MT system built on Moses (Koehn et al., 2007),
- a transfer-based deep-syntactic MT system TectoMT (Popel and Žabokrtský, 2010),
- and a rule-based post-editing tool Depfix (Rosa et al., 2012).

The phrase-based core of Chimera is a fairly standard PBMT system with several tweaks. We add the translations produced by TectoMT into the phrase-based system through an additional phrase table extracted from synthetic data. Finally, the output of Moses is post-edited using Depfix. We will describe each part in more detail in the following sections.

System Versions

Before describing the individual components, let us first note that Chimera went through slight changes during the four years of participating in WMT. Most importantly, it competed in the *unconstrained* track for the first three years. This mainly allowed us to use large amounts of in-domain monolingual Czech data. In 2015, we also added a significant amount of parallel data, most of which was later included in CzEng 1.6 (Bojar et al., 2016b), the official constrained parallel corpus for WMT16.

In 2016, we constrained the training data for the first time to allow for a better comparison with competing systems. This implies that our monolingual training data were limited.

Additionally, we experimented with various "tweaks" each year. In some cases, these tweaks were included in our primary submission.

For simplicity, we only describe the two versions of Chimera directly related to this work: our submission from 2015 (unconstrained) because this is the system that we analyze most thoroughly here (building on Tamchyna and Bojar 2015), and 2016 (constrained) which is the final baseline for our work on discriminative models.

Phrase-Based Component

The PBMT component of Chimera differs from a "vanilla" phrase-based system in several aspects. We use factored translation (Koehn and Hoang, 2007)

and translate English surface word forms jointly into Czech word forms and their morphological tags (and sometimes even lemmas).

This factored scheme makes our target-side data somewhat more sparse. There is a high amount of syncretism in Czech; a single word form often represents multiple morphological variants. For example, adjectives in the inflectional paradigm "jarní" do not inflect at all: the same word form is used for all (3-4) genders, 7 cases and two numbers (singular and plural).

Empirically, we find that once the training data is large enough, attaching the morphological tag to the surface form does not harm the BLEU score very much, and the improvements from the additional models enabled by this scheme easily outweigh this potential slight loss.

Capitalization or upper-casing of surface forms is determined using the "supervised truecase" scheme: as part of pre-processing, we lemmatize and tag all of our parallel data. We then use the letter case of the lemma to determine the case of the surface form. We will denote this factor stc from now on.

We evaluated several possible ways of handling capitalization and upper case in Bojar et al. (2013b); unsupervised truecasing, using a recaser and translating directly to surface forms all performed worse than our stc scheme.

We use the positional morphological tagset of Czech which is described more thoroughly in Hajič and Vidová-Hladká (1998).

Thanks to the factored configuration, the system has access to Czech morphological tags on the target side during translation. This allows us to build language models (LMs) not only over the surface forms but also over the sequences of tags. This should encourage our MT system to produce outputs which are morpho-syntactically more coherent.

Training Data

Let us describe the training data for both years of Chimera (2015 and 2016) in more detail.

Training data for 2015 are summarized in Tables 3.1 and 3.2, which are adapted from Bojar and Tamchyna (2015).

The constrained part of our parallel data consists of CzEng 1.0 (Bojar et al., 2012) and Europarl (Koehn, 2005). We then add OpenSubtitles corpora as collected and processed by OPUS,[1] our own pre-processing of DGT Acquis[2] which includes sentence alignment (Varga et al., 2005) and de-duplication, and

[1] http://opus.lingfil.uu.se/

[2] https://ec.europa.eu/jrc/en/language-technologies/dgt-acquis

Source	# sents	# en tokens	# cs tokens	Constrained?
CzEng 1.0	14.83M	235.19M	206.05M	✓
Europarl	0.65M	17.62M	15.00M	✓
OpenSubtitles	33.25M	291.38M	237.61M	-
DGT Acquis	3.82M	93.44M	84.81M	-
EAC-TM	3351	24330	23106	-
ECDC-TM	2499	4092	41591	-

Table 3.1: Chimera 2015: Summary of parallel data used in our constrained and full setup.

			Full				Constrained		
	# sents	# tokens	l	b	m	lm	l	m	lm
Czech Press	305.41M	4852.59M	-	✓	-	-	-	-	-
CWC articles	38.42M	627.97M	-	✓	-	-	-	-	-
CzEng news	0.20M	4.22M	-	✓	✓	✓	-	✓	✓
RSS	4.81M	73.68M	✓	✓	✓	✓	-	-	-
WMT mono	44.08M	738.88M	✓	✓	✓	✓	✓	✓	✓

Table 3.2: Chimera 2015: Monolingual data sources and LMs, l=long, b=big, m=morph, lm=longm.

finally, we also include two small translation memories provided by the EU, namely EAC-TM[3] and ECDC-TM.[4]

Our monolingual data do not overlap with the parallel data very much: we only use the small news section of CzEng 1.0. We add all WMT News Crawl corpora from 2007 until 2014. We use the Articles section of the Czech Web Corpus (CWC, Spoustová and Spousta 2012). Czech Press is a large in-house collection of news. Finally, RSS is the output of Ondřej Bojar's collection script which stores RSS feeds of major Czech press websites and which is updated every day.

For 2016, our data is simpler as we only use the constrained datasets. CzEng 1.6 in the pre-release version (the official training set for WMT) contains 51.34 million sentence pairs with 713.5 and 605.0 million tokens in English and Czech, respectively.

Monolingual data statistics are summarized in Table 3.3. In this constrained scenario, all LMs use the same training data.

[3] https://ec.europa.eu/jrc/en/language-technologies/ecdc-translation-memory
[4] https://ec.europa.eu/jrc/en/language-technologies/eac-translation-memory

	# sents	# tokens
CzEng 16.pre news	0.21M	4.52M
WMT mono	51.47M	871.15M

Table 3.3: Chimera 2016: Monolingual data sources.

Language Models

In 2015, we used four LMs in the Chimera system: "long", "big", "morph" and "longm"; see Table 3.2.

The *long* LM is a 7-gram model over surface forms (stc) trained mainly on WMT News Crawl data. We treat each year of the News Crawl as a separate corpus; we estimate a LM over each of them and then use SRILM interpolation (Stolcke, 2002) to combine the individual models. The interpolation minimizes the perplexity of a held-out data set; we use older WMT dev-sets for this purpose.

The *big* LM is a model over stc which uses as much monolingual data as possible. We were not able to build a 5-gram model due to disk requirements so we reduce the LM order to 4-grams.

The *morph* and *longm* are LMs trained over morphological tags. Tags are naturally much less sparse than word forms; there are only around two thousand distinct tags in our corpora. This enables us to effectively model larger sequences; the *morph* is a 10-gram LM over tags and *longm* is a 15-gram LM. In practice, *longm* does not bring measurable improvements over *morph*.

In 2016, we simply took over the scheme from the previous year, even though the constrained setting could possibly have benefited from a different combination. All of our LMs are trained using all the monolingual data available in the constrained setting. We again have *long*, *big*, *morph* and *longm* with roughly the same parameters (except for pruning of low-frequency n-grams). The *long* model is again the only interpolated model. In comparison with 2015, we also add a 4-gram model over word lemmas.

TectoMT

TectoMT (Popel and Žabokrtský, 2010) is a translation system which uses the analysis-transfer-synthesis approach. It is loosely based on the Functional Generative Description formalism (Sgall, 1967); it uses the tectogrammatical (deep syntactic) layer as the transfer layer.

Each input sentence is first automatically analyzed; after tokenization and some normalization heuristics, TectoMT runs a tagger (Hajič et al., 2007) and obtains a dependency parse. MST parser (McDonald et al., 2005) is used which allows for non-projective trees. The (shallow) syntactic tree corresponds to the *analytical* layer. TectoMT then applies a set of rules and models which further analyze the sentence on the *tectogrammatical* layer.

Roughly speaking, nodes in the tectogrammatical tree correspond to content words. Each node is represented by its tectogrammatical lemma (*t-lemma*), *functor* (semantic role in the sentence; e.g. actor, patient, predicate), and *grammatemes* (meaning-bearing grammatical attributes, such as the semantic part of speech, tense or number).

Additionally, *formemes* are stored in each t-node which provide a useful concise representation of relevant morpho-syntactic properties on the t-layer. They allow the transfer module to easily access information which is otherwise only available on the level of shallow syntax (a-layer).

The basic assumption in TectoMT transfer is that t-trees are so abstract that they are *isomorphic* in English and Czech: both the English and Czech t-trees have the same number of nodes and identical structure. This simplifies transfer considerably: we already have the deep tree, the only task is to correctly label its nodes. A maximum-entropy classifier is used to select the best translation and formemes locally for each node.

Once transfer is complete, TectoMT has a tectogrammatical tree in the target language. A pipeline of generation rules is then applied to *realize* this deep-syntactic structure on the surface. Naturally, generation moves in the opposite direction than analysis; first, the t-tree is transformed into an a-tree. The a-tree is then linearized and words are inflected and post-processed to create the final translation.

Because the generation pipeline is based on rules, TectoMT tends to produce grammatically correct outputs where phenomena such as long-distance agreement (e.g. between verbs and subjects) are handled correctly by design. This implies that while TectoMT often makes severe errors in MT, these errors are very different from the typical problems of statistical phrase-based engines. TectoMT is therefore is an almost ideal candidate for system combination.

For completeness, let us note that TectoMT today is one of the applications of the underlying modular NLP toolkit Treex.[5]

[5] `http://ufal.mff.cuni.cz/treex`

Integration in Chimera

The approach to integration of TectoMT is simple yet surprisingly effective. Bojar et al. (2013b) obtain translations of the dev-set and the test-set from TectoMT. They then view the pairs of source sentences and their translations as a new parallel corpus. They word align this corpus and extract phrasal translations from it using the standard Moses pipeline. They then add the new synthetic phrase table as an additional model in the log-linear combination.

MERT is then used to balance out the importance of the standard phrase table (obtained from true parallel data) and the TectoMT phrase table. Note that MERT can tune the weights reliably only because TectoMT translations of the dev-set are also included in the phrase table; otherwise, the synthetic phrase table would only be used for the most common words or in cases of overlap between the dev- and test-data.

This approach is dubbed by the authors "poor-man's system combination". It is somewhat cumbersome – whenever a new document is to be translated, the TectoMT phrase table needs to be re-created. On the other hand, it is also surprisingly effective as it allows to combine Moses and TectoMT translations under *any* phrasal segmentation. The phrase-based component is also not limited to its 1-best or n-best outputs but can instead explore its full search space, enriched with TectoMT translation suggestions.

Depfix

Depfix (Rosa et al., 2012) is a rule-based system for automatic post-editing of MT outputs. It is tailored to English-Czech translation, but porting it to other language pairs could be relatively straightforward.

It operates by applying a sequence of rules on the MT output. It uses morphological and syntactic information both from the source sentence and the target sentence. Rosa et al. (2012) even implemented their own version of the MST parser adapted to MT outputs. Depfix also uses the word alignment between the source and translation to determine word correspondences.

Depfix rules are hand-tailored to fix some typical and systematic errors made by MT systems. Morphological agreement, the correct transfer of grammatical number, tense or negation are all handled by the system.

While Depfix usually has only a negligible effect on automatic evaluation scores (BLEU), it does improve MT output quality as judged by human annotators, according to the WMT shared task results (Bojar et al., 2013a, 2014).

	System	BLEU	TER	Manual
WMT13	M+T+D	20.0	**0.693**	**0.664**
	M+T	**20.1**	0.696	0.637
	Google Translate	18.9	0.720	0.618
	T	14.7	0.741	0.455
WMT14	M+T+D	21.1	0.670	**0.373**
	uedin-unconstr.	**21.6**	**0.667**	0.357
	M+T	20.9	0.674	0.333
	Google Translate	20.2	0.687	0.168
	T	15.2	0.716	-0.177
WMT15	M+T+D	**18.8**	**0.715**	**0.686**
	Google Translate	16.4	0.750	0.515
	uedin-jhu	18.3	0.719	0.503
	T	13.4	0.763	0.209
WMT16	udedin-nmt	**26.3**	**0.639**	**0.59**
	M+T+D	21.7	0.677	0.30
	Google Translate	23.2	0.678	0.19
	T	15.2	0.730	-0.03

Table 3.4: Automatic scores and results of manual ranking in WMT 2013-2016. BLEU (cased) and TER from `matrix.statmt.org`. The top other system and Google Translate reported for reference.

When running Chimera, we simply translate each input sentence using Moses with the factored LMs and the added TectoMT phrase table. We then post-process the outputs using Depfix as a final step.

3.2 Analysis of the Combination

Chimera was analyzed thoroughly in Tamchyna and Bojar (2015). In this section, we look at some of the results from that article and interpret them in relation with the main goal of our work. All of these analyses were carried out with Chimera 2015 as described above.

Automatic and Manual Evaluation

To illustrate the overall performance of Chimera and its components, we provide the official results of WMT news translation tasks from 2013 to 2016. Table 3.4 lists the results. Letters M, T, D stand for Moses (here: the phrase-based component of Chimera), TectoMT and Depfix, respectively.

	Constrained	Full	Delta
M	21.28	22.59	1.31
M+T	23.37	24.24	0.87
Delta	2.09	1.65	-

Table 3.5: BLEU scores on WMT newstest2014 of the first two components of Chimera 2015. Adapted from Bojar and Tamchyna (2015).

(r)	obývací zóna s jídelní a kuchyňskou částí v domácnosti mladého páru .
	living zone with dining and kitchen section in household young$_{gen}$ couple$_{gen}$.
(m)	obývací zóna s jídelnou a kuchyní **v sekci** domácnosti **mladý pár** .
	living zone with dining_room and kitchen **in section** *household$_{gen}$* **young$_{nom}$ couple$_{nom}$** .
(t)	**živá** zóna **pokoje** s jídelnou a s kuchyňským oddílem v domácnosti mladého páru .
	alive *zone* **room$_{gen}$** *with dining_room and with kitchen section in household young$_{gen}$ couple$_{gen}$.*
(c)	obývací prostor s jídelnou a kuchyní v domácnosti mladého páru .
	living space with dining_room and kitchen in household young$_{gen}$ couple$_{gen}$.

Figure 3.1: Example of translations by various stages of Chimera. The input sentence: "*the living zone with the dining room and kitchen section in the household of the young couple .*" Table rows correspond to: (r) reference, (m) Moses, (t) TectoMT, and (c) Chimera (without Depfix). Errors are in bold, glosses are in italics. Adapted from Tamchyna and Bojar (2015).

We list the results here mainly to illustrate that Chimera is indeed a very strong system and even at the time of WMT16, it performs better in terms of human evaluation than Google Translate, perhaps the best-known publicly available MT system.

The final combination is better than either component by itself. Table 3.5 illustrates the difference in BLEU that TectoMT provides over plain Moses. The table also shows that the improvement holds when we go from the constrained setting to the unconstrained system. TectoMT is a source of grammatically coherent translations possibly with unseen morphological variants and this advantage does not disappear as we add (very large) additional training data to the statistical component. These results again illustrate that Chimera is a very strong baseline due to its hybrid nature.

For completeness, we re-use the example translations from Tamchyna and Bojar (2015) to illustrate how the individual components can often work together to improve the final translation. Plain Moses (m) generates a relatively adequate translation with two problems:

- The syntactic attachment of the word "section" is wrong in the translation.
- The phrase "young couple" is in the wrong case (nominative).

TectoMT outputs a fully grammatical translation with serious lexical translation errors at the beginning of the sentence; the translation of "living" roughly means "alive" in Czech, there is also a superfluous word "pokoje".

The full combination fixes both the errors of Moses and of TectoMT. For the second error made by Moses ("young couple"), the system presumably uses the correct translation provided by TectoMT.

Experiments with Forced Decoding

Forced decoding refers to running inference in a model when the desired output is already known. This procedure is useful when we want to obtain the model score (the probability according to the model) of some given output, or to determine whether the model is even capable of producing the output.

<table>
<tr><th>all</th><th>different?</th><th>reachable?</th><th>score diff</th></tr>
<tr><td rowspan="3">3003</td><td rowspan="2">2665</td><td rowspan="2">1741</td><td>1601 (<)</td></tr>
<tr><td>140 (>)</td></tr>
<tr><td>924</td><td>(unreachable)</td></tr>
<tr><td></td><td>338</td><td colspan="2">(identical)</td></tr>
</table>

Table 3.6: Forced decoding – an attempt of M to reach the test set translations produced by M+T (Tamchyna and Bojar, 2015).

In Tamchyna and Bojar (2015), we attempted to find out whether the Moses component could produce the translations of M+T by itself and if so, why it did not. We report this analysis here because it helps illustrate how much we can improve the results of Moses by building a better model: in the previous section, we saw that TectoMT provides correct (and sometimes novel) morphological variants of words.

These experiments attempt to find whether a better model within Moses (such as our discriminative model) can theoretically lead the phrase-based component to these better translations without the need for TectoMT. Or alternatively, whether TectoMT indeed provides so many novel word forms that better modelling of translation has no hope of reaching these results.

Table 3.6 shows the results of the forced decoding experiment. Out of the full test set (3003 sentences), M and M+T produced identical translations in 338 cases. From the remainder, roughly two thirds of translations by M+T

were in fact reachable for plain Moses. The analysis shows that model score is the main problem: translations by M+T have a higher BLEU score than plain Moses outputs (24.78 vs. 23.03) but they mostly (in 1601 cases) have a lower model score (labeled as "<" in Table 3.6).

In 140 cases, when we guided the decoder towards the translation of M+T, we obtained a higher model score than that of the original translation. These are search errors and we nearly eliminated them by adjusting parameters of the search (specifically, we increased the pop limit of the cube pruning algorithm to 5000), reducing the number of such cases to 28 sentences.

Overall, this is a promising finding: while not all of the better translations are reachable for the statistical component, most of them can in fact be produced by Moses itself. If we add new model components to Moses which better capture translation quality, we can potentially guide the model towards them and obtain a higher BLEU score.

Errors in Morphology

The overall goal of our work on discriminative models is to improve lexical choice and morpho-syntactic coherence of translations. The analysis from Tamchyna and Bojar (2015) on morphological errors provides a potentially useful insight into which types of errors in morphology are the most prominent. We can determine (i) which phenomena are improved by adding the transfer-based system and also (ii), which phenomena resist even the combination.

We looked at the WMT14 test set. We lemmatized the system outputs as well as the reference translation and then used a monolingual word aligner based on hidden Markov models (Zeman et al., 2011) to find the corresponding tokens in the reference and the outputs. This procedure matched a different number of lemmas in each setting.

We then looked at each system variant independently and automatically analyzed the matched tokens. We look at cases where the lemma matches but the surface form differs – we consider these to be *morphological errors*. The system found the correct lemma but failed to inflect it as needed. This is of course a simplified view: a different wording elsewhere in the MT output may have required this different form and vice versa. Even so, this analysis gives us interesting insights when we look at statistics from the full test corpus. We limit ourselves to the POS that inflect in Czech: adjectives (A), numerals (C), nouns (N), pronouns (P) and verbs (V).

Because plain Moses is the weakest system, it matched a lower number of lemmas with the reference (39255, see Table 3.7). We only consider the matched

System	# lemmas	# errors	# errors by part of speech				
			A	C	N	P	V
M	39255	**5877**	1200	90	2727	502	1358
M+T	**39684**	5382	1066	75	2454	480	1307
M+T+D	**39610**	5369	1071	76	2431	468	1323

Table 3.7: Morphological errors made by Chimera divided by part of speech. A=adjective, C=numeral, N=noun, P=pronoun, V=verb (Tamchyna and Bojar, 2015).

lemmas when looking for errors in morphology, so this system has an advantage over the other variants which matched around 400 more lemmas (and therefore had more "opportunities" to make an error in our simplified counting). However, plain Moses also makes the largest number of errors in morphology: 5877 as opposed to 5382 and 5369.

We can make interesting observations when we further look at the distribution of errors across different parts of speech (POS). Naturally, the distribution is very similar to the frequency of the POS in the data. For instance, roughly 46% of the errors (2727/5877) are made in nouns; nouns make up about 37% of the tokens in the test set (limited to A, C, N, P and V).

However, the reduction of errors when we add TectoMT is interesting. There is only little difference in verbs (the number of errors is only 4 per cent lower) and pronouns (5%). On the other hand, we see a large reduction in nouns and adjectives (about 11%).

We can conclude that verbs and pronouns are frequent and difficult; not only for Moses but also for the combination with the deep-syntactic TectoMT.

We further look specifically at errors of plain Moses which M+T fixed. When we focus on nouns and adjectives, we find that in almost all instances, the morphological tags differ only in *case* (393 out of 407 errors fixed). This implies that even with the morphological LMs, Moses struggles with phenomena related to morphological coherence, such as valency and noun-adjective agreement. Fixing the prediction of morphological case seems to be a promising direction.

3.3 Limitations of the Phrase-Based Component

So far, we have illustrated how Moses and TectoMT work together in combination. However, a simpler and less language-dependent solution would be to avoid using TectoMT and make the model in Moses as adequate as possible.

In this section, we show that phrase-based models have some inherent limitations. We describe these problems and illustrate them on several examples of translations produced by the plain Moses component of Chimera.

Data Sparsity in the Translation Model

Recall that during translation, the phrase-based system looks up possible translations of all source spans in the phrase table and then searches for their best possible combination according to the log-linear model.

In an ideal situation, we can translate the input sentence using only a couple of long phrases. This is desirable because we can be reasonably sure that morphological agreement and syntax within the phrasal translation is correct: we saw this translation in the training data. On the other hand, when we build the translation from very short phrases or even word by word, we risk making an error in coherence at every step.

In practice, phrase-based systems typically use phrases only a couple of words long. In Chimera, the average phrase length when translating WMT news tests is only around 2.5 words (Tamchyna and Bojar, 2015).

This issue is an example of *data sparsity*: because the model needs to observe each phrase in the data, we would in principle require exponentially larger training data to obtain a linear improvement in the average length of applied phrases. Note that we would also need to observe each source phrase many times with different translations, in order to obtain robust estimates of its phrasal translation probabilities.

Data Sparsity in LMs

In reality, the system is therefore often forced to translate the input in short chunks. In this situation, it relies even more on the language model (LM) to "knit" the short phrasal translations together. Unfortunately, LMs suffer from data sparsity as well, for similar reasons: conventional (non-neural) LMs estimate the conditional probability of the next word given a context from n-gram *counts* in the training data.

We would ideally like to model as long context as possible. But with every added word, we again need exponentially larger training data to robustly estimate n-gram probabilities. LMs work well only when the information required to disambiguate the next word is *locally* available in a very small window. Moreover, when the immediate context is very uncommon (and therefore unobserved in the training data), LMs fall back to low-order n-grams or even

unigram probabilities. In this case, LMs fail even when the required information is close.

Context Information

Finally, let us go back to the phrase table and make one last remark. The phrase-based model makes strong independence assumptions when estimating translation probabilities. One of them is that the conditional probability of a phrase $\bar{f}$ being translated as $\bar{e}$ is independent of all other words in the source sentence. We will show that in combination with data sparsity, this leads to clearly wrong translations.

Translation Examples

Consider Figure 3.2 which shows translations produced by the statistical component of Chimera (plain Moses). In the first set of examples, we show the role of source context in the disambiguation of word sense.

The English word "shooting" is ambiguous. When translating into Czech, we need to know which sense of the word is meant in the current sentence. For simplicity, let us consider only two basic meanings: shooting a weapon (translated into Czech as "střelba") or shooting a film (Czech translation is "natáčení"). In the sentences in Figure 3.2, the latter sense is correct. The cue which helps us decide is the English word "film".[6]

In the first sentence, the phrase-based system produces the correct translation. This is possible because (i) the phrase table may simply contain the whole phrase "shooting of the film", or (ii), the LM probability of "natáčení filmu" is sufficiently large. Once we move the cue word outside of the immediate context by inserting the word "expensive" inbetween, the data sparsity and strong independence assumptions of the model break the translation: the word "shooting" is translated in the wrong sense as "střelby".

The phrase table cannot cover "shooting" and "film" by a single phrase any more and phrasal translation probabilities do not capture the context of the phrases. The LM has a similar problem: the words "střelby" and "film" are simply too distant. Note that if we only consider bigram probabilities, the whole translation is fine: "střelby na", "na drahý", and "drahý film" are all plausible n-grams.

[6]The sentence could hypothetically also describe firing a weapon at a copy of an expensive film. While possible, this meaning is a priori improbable and an intelligent translation system should work with the default reading, when extra-sentential context which could provide more evidence is not available.

Input	Chimera Output	
shooting of the film .	natáčení filmu . ***shooting***$_{camera}$ *of_film .*	✓
shooting of the expensive film .	střelby na drahý film . ***shootings***$_{gun}$ *at expensive film .*	×
the man saw a cat .	muž uviděl kočku . *man saw* ***cat***$_{acc}$.	✓
the man saw a black cat .	muž spatřil černou kočku . *man saw* ***black***$_{acc}$ ***cat***$_{acc}$.	✓
the man saw a yellowish cat .	muž spatřil nažloutlá kočka . *man saw* ***yellowish***$_{nom}$ ***cat***$_{nom}$.	×

Figure 3.2: Examples of problems in the phrase-based component of Chimera: lexical selection and morphological coherence over larger context. Each translation has a corresponding gloss in italics.

In the next set of examples in Figure 3.2, we focus on morphology instead of lexical choice. Aside from selecting the correct lexical translation (lemma), the MT system needs to output the correct surface form of the words.

Consider the sentence "the man saw a cat .". Because it has a very simple structure and all words are sufficiently common, the phrase-based system correctly inflects all word forms. In particular, it correctly selects the accusative case for the Czech word "kočku" (cat). When we insert an adjective between the verb and its object, the system still handles it well as long as the adjective itself is common enough. If we "surprise" the system by choosing an unusual adjective, the n-gram context breaks again and the whole noun phrase is mistranslated into nominative case. Note that the correct inflection of the adjective "nažloutlý" was available as a translation of "yellowish" in the phrase table.

The correct morphological case cannot be reliably inferred from the source context only. Source context information can certainly help in the decision – for instance, English subjects are often translated into nominative case in Czech, direct objects in Czech are often accusative etc. However, there are many phenomena that affect morphology which are only present on the target side: different translations of the same English verb can have different valency requirements; the MT system can switch between active and passive voice, reversing the original roles of subject and object in the sentence etc.

Consider again the second set of examples in Figure 3.2. Instead of translating "saw" as "uviděl", the decoder could have chosen the translation "všiml

si". In this case, the object needs to be in the genitive case. This information is simply not available on the source side.

The examples clearly show that considering wider context information can be beneficial for translation quality. In the first sentence, a model which can effectively incorporate the cue word "film" when disambiguating the translation of "shooting" should be able to select the correct translation even when the two words are somewhat far apart.

In the second sentence, the model should be able to recognize which verb appeared in the sentence and select the morphological case of the object according to its valency requirements.

4

Machine Learning Background

In this chapter, we describe the basic machine learning (ML) notions relevant for this thesis, especially linear models (linear and logistic regression) and their training.

We first describe the theoretical concepts underlying the models. We then move on to describing the implementation that we selected for our work, namely the Vowpal Wabbit toolkit.[1]

4.1 Theoretical Foundations

In order to formalize our work, we first define some basic notions. Let $D = (X, Y)$ be the training data, where $X = (x_1, \ldots, x_n)$ are the inputs and $Y = (y_1, \ldots, y_n)$ the associated outputs.

In this very general formulation, the goal of ML models is to learn from the training data to predict the correct value y_i given the instances x_i.

One branch of models deals with *classification* problems: the usual scenario for classification is that y_i can be either *positive* or *negative* (binary classification); alternatively, y_i can be one of multiple classes (multi-class classification).

For example, we can build models to answer questions such as "Is this sentence written in German?" (binary decision). Our inputs x_i are then sentences (written in German or other languages) and desired outputs y_i are either 0 or 1 (or -1 and +1).

[1] `http://hunch.net/~vw`

Alternatively, we can ask the question "In which language is this sentence written?". Our inputs x_i are the same as in the previous case but y_i is now selected from a pre-defined set of possible values (e.g. German, English, etc.).

Another branch are *regression* models. In this case, values y_i are usually assumed to be real numbers. An example of regression would be, given a house, to predict its price on the real-estate market.

Linear Models

Let us assume that we have a function $\text{fv}(x_i)$ that generates a vector of features given our input x_i. Going back to the example of language classification, our function could return a vector which contains the occurrences of various characters, their bigrams (pairs or consecutive characters) etc.

We can then attempt to capture the relationship between our input x_i and the predicted value y_i with a linear model. Let us denote the predicted value $\hat{y}_i$. Then the basic linear model is defined as follows:

$$\hat{y}_i = \boldsymbol{w} \cdot \text{fv}(x_i) \tag{4.1}$$

The prediction of our model is simply the dot product between a vector of *feature weights* $\boldsymbol{w}$ and the feature vector which describes the current input.

We assume that there is a linear relationship between $\text{fv}(x_i)$ and y_i, i.e. that the value y_i is a linear combination of feature values.

Training a linear model corresponds to finding such a vector of feature weights that minimizes the error which our model makes. Consider again the task of classifying whether a sentence in written in German. We would like our model to assign a high weight to features which correspond to occurrences of characters which best identify German (such as "ß"). That way, when these characters occur at test time, the high positive weight will increase the value of the dot product and the model will output a positive prediction.

Linear Regression

The basic form of the linear model that we defined in Equation 4.1 is a linear regression model.

Parameter Estimation

Unless stated otherwise, we always assume that each training instance is drawn independently and that the training instances are identically distributed (the "i. i. d." assumption).

Under an additional assumption about the distribution of target values in the training data, it can be shown that maximum likelihood estimation (MLE) of model parameters $\boldsymbol{w}$ for linear regression corresponds to minimizing the squared loss (Bishop 2006, section 1.2.5):

$$\mathcal{L}(\boldsymbol{w}) = \sum_i (y_i - \boldsymbol{w}\,\mathrm{fv}(x_i))^2 \tag{4.2}$$

Linear Regression for Classification

Note that in general, the values of linear regression predictions (the dot product) lie between $-\infty$ and $+\infty$. For (binary) classification, we would ideally like the model to output values around 0 for negative examples and +1 for positive examples.

This fact does not prevent us from using the simple dot product in practice for classification. When we label our training instances with 0 and 1, minimizing the squared error will lead to a model which in practice produces predictions *roughly* in this range.

Logistic Regression

A more principled solution is to view the classification problem in a probabilistic setting.

Let us define several notions first. A *discriminative model* is an estimate of the *conditional* probability distribution $P(Y|X)$. On the other hand, a *generative models* attempts to estimate the *joint* probability distribution $P(X, Y)$.

Because $P(X, Y) = P(Y|X) \cdot P(X)$ (Bayes' rule), the common intuition is that discriminative models are preferable for classification because they directly solve the task at hand whereas generative models learn a more complex task (see e.g. Bishop 2006, section 1.5.4). (The advantages of generative models include the ability to create additional data instances by sampling the joint distribution and outlier or novelty detection, through estimating $P(X)$).

Logistic regression is a simple probabilistic discriminative model. In its basic form, it is a binary classifier – training examples are either *positive* or *negative* and the classifier produces the probability that an example is positive, $P(1|x)$.

Its mathematical formulation is very simple as well – it can be viewed as linear regression (a basic linear model) squeezed by the logistic sigmoid function to ensure that its predictions are valid probabilities; the range of logistic sigmoid is $(0, 1)$.

Assuming that we already have a weight vector w, the output of logistic regression is defined as follows:

$$P(1|x) = \frac{1}{1 + \exp(-\boldsymbol{w}\,\text{fv}(x))} \tag{4.3}$$

We can turn this conditional probability into a classification decision simply by choosing a threshold (typically 0.5) and declaring an example as positive when the value exceeds this threshold.

Parameter Estimation

Using maximum likelihood estimation to train our model means that we maximize the probability of the data given the model parameters. The probability of the data in the case of logistic regression can be expressed as follows (assuming $y = 1$ for positive examples and 0 for negative examples):

$$P(D|\boldsymbol{w}) = P(Y, X|\boldsymbol{w}) = \prod_i P(1|x_i, \boldsymbol{w})^{y_i}(1 - P(1|x_i, \boldsymbol{w}))^{1-y_i} \tag{4.4}$$

Note that whenever $y_i = 1$, the first term (the probability of the positive class) is unchanged and the second term equals one, and vice versa. We are interested in minimizing the cross-entropy:

$$\mathcal{L}(\boldsymbol{w}) = -\log P(D|\boldsymbol{w}) = -\sum_i \log P(1|x_i, \boldsymbol{w})^{y_i}(1 - P(1|x_i, \boldsymbol{w}))^{1-y_i} \tag{4.5}$$

In order to optimize this objective function, we need the formula for the gradient with respect to the model parameters:

$$\nabla\mathcal{L}(\boldsymbol{w}) = -\sum_i \text{fv}(x_i)\left(y_i - \frac{1}{1 + \exp(-\boldsymbol{w}\,\text{fv}(x_i))}\right) \tag{4.6}$$

Multiple Classes

The standard practice for multi-class classification is to use multiple logistic regression, also known within NLP as the maximum-entropy classifier. In our work, we instead rely on the VW reduction schemes which address multi-class (or even multi-label) classification by combining many binary classifier models.

Let us only note here that the generalization of the logistic function for multiple classes is the *softmax* function. Suppose we have K possible labels

(classes) $\{1, \ldots, K\}$. Let us also generalize our feature vector function so that its output also depends on the currently predicted class $\hat{y}$; i.e., we have $\text{fv}(x, \hat{y})$. Then the softmax represents the probability of the example x belonging to the predicted class $\hat{y}$:

$$P(\hat{y}|x) = \frac{\exp(\boldsymbol{w}\,\text{fv}(x, \hat{y}))}{\sum_{y'=1}^{K} \exp(\boldsymbol{w}\,\text{fv}(x, y'))} \tag{4.7}$$

4.2 Implementation

We use Vowpal Wabbit (VW) for most of our work. VW is a fast ML toolkit with support for various types of models, distributed computation and on-line learning. Here we briefly describe the parts of VW relevant for this thesis.

Feature Hashing

First, let us discuss a more technical aspect of VW which is useful for understanding how the model parameters are represented in the toolkit.

For efficiency, VW does not work with feature strings. Instead, each feature has an integer identifier from a pre-defined range. The user defines how many bits to use for storing the feature IDs (i.e., the upper limit on how many distinct features can be used). For example, with 12 bits, the number of possible distinct features is $2^{12} = 4096$.

When the user provides a feature name as a string, the name is *hashed* into this pre-defined range and the hash becomes the feature ID. This allows VW to store data in a compact way and to maintain the weight vector as a single, dense array, as opposed to a sparse data structure. (The user can also provide feature IDs as integers; in this case, feature hashing is omitted by default.)

By default, VW does no checking for hashing collisions. It is again up to the user to define a sufficiently large hash size so that collisions are mostly avoided.

Training Criteria

VW does not directly implement the various ML models (such as logistic regression) as separate modules. Instead, it operates with a basic linear model (i.e., the scalar product between a feature vector and a single weight vector) and provides several "loss functions" which turn this model into different classi-

fiers, e.g. logistic regression (logistic loss), support vector machine (hinge loss), or linear regression (squared loss).[2]

For our work, we experiment with logistic loss and squared loss which we introduce here for completeness:

$$\mathcal{L}_{logistic}(\boldsymbol{w}, x, y) = \log(1 + \exp(-y \cdot \boldsymbol{w} \cdot \text{fv}(x)) \tag{4.8}$$

$$\mathcal{L}_{squared}(\boldsymbol{w}, x, y) = \frac{1}{2}(\boldsymbol{w} \cdot \text{fv}(x) - y)^2 \tag{4.9}$$

Multi-Class Classification

VW does not implement multi-class classification (such as multiple logistic regression) directly. Instead, it provides reductions to binary classification or regression problems. In our work, we use the cost-sensitive one-against-all reduction for training models with logistic loss.[3]

In this setting, VW internally creates K binary classification problems where K is the number of possible classes. Each classifier decides whether the current instance belongs to the given class. Additionally, each label has an associated cost which can be used to weigh misclassifications differently or to allow for multi-label classification (where multiple labels can be correct for a single instance).

For squared loss, we use VW in a mode which converts each multi-class decision into K independent regression problems. Each regressor directly predicts the cost associated with the given label.[4]

Label-Dependent Features

When the number of possible classes becomes too large, multi-class classification tends to perform poorly – if there are K possible classes, we would effectively need to train K independent models, each predicting a single class.

Notice how we define multiple logistic regression in Section 4.1: we extended our feature vector function to take the predicted class as an additional parameter. Some ML toolkits are designed to keep K independent vectors of feature weights and select the vector based on the currently predicted class. K separate models are therefore trained.

[2] `https://github.com/JohnLangford/vowpal_wabbit/wiki/Loss-functions`

[3] The exact command-line setting is "`csoaa_ldf mc`".

[4] The setting in this case is "`--csoaa_ldf m`", see e.g. `https://www.umiacs.umd.edu/~hal/tmp/multiclassVW.html` for a thorough discussion.

VW internally uses a slightly different (and more flexible) approach: when predicting label k, it combines the hashes of active features and the label ID into a new hash (conceptually, this is similar to concatenating the feature string and the label).

In the *label-dependent features* mode, VW does not automatically do this combination and instead relies on the user to *explicitly* make the features class-dependent. Each class is therefore defined not by its label but by the features associated with it.

This approach has several advantages: class labels are not necessary and we need not specify the number of classes ahead of time. More importantly, we can decide that some features should in fact not be combined with the label and should instead be *shared* among multiple classes.

This flexibility is advantageous because often classes (labels) are not completely unrelated and some features should in fact be shared among them. This can lead to more robust parameter estimation because data sparsity is reduced: before, we effectively updated a separate feature vector depending on which class was predicted (i.e., we had an independent model for each possible class). With shared features, we may observe the same feature in multiple classes and get more reliable statistics for it.

For illustration, consider again our language identification example from Section 4.1. When we know that a sentence is in German, we also know that it is written in Latin script and uses some characters more frequently than other languages (letters with umlauts, "ß" etc.). When a sentence is written in Russian, it uses the Cyrillic script. But Cyrillic is also used in other related languages, such as Ukrainian or Bulgarian.[5] Because the script is shared, it can become a target-side feature: for the languages which use it, we can define a new feature "cyr" and combine our input features with this label as well. Similarly, we can add "lat" for languages written in Latin script.

Let us consider a very simple example. Suppose we only classify five languages: English, German, Russian, Bulgarian and Ukrainian, and that we only have single-character features. For the Russian training sentence "спасибо" ("thank you"), the instance would be defined as follows with label-dependent features:

```
0 en_с lat_с en_п lat_п en_а lat_а en_с lat_с en_и lat_и ...
0 de_с lat_с de_п lat_п de_а lat_а de_с lat_с de_и lat_и ...
1 ru_с cyr_с ru_п cyr_п ru_а cyr_а ru_с cyr_с ru_и cyr_и ...
```

[5] These languages all use different variants of Cyrillic but nonetheless share the majority of character types.

```
0 uk_c cyr_c uk_п cyr_п uk_a cyr_a uk_c cyr_c uk_и cyr_и ...
0 bg_c cyr_c bg_п cyr_п bg_a cyr_a bg_c cyr_c bg_и cyr_и ...
```

This setting allows the model not only to learn how to recognize Russian. It can also learn that Cyrillic characters (such as "п") should not appear in languages written in Latin script (features such as "lat_п" should receive a large negative weight) and vice versa (from English and German training examples). When the model encounters a sentence written in Cyrillic at test time, languages which use Latin script (and the "lat_?" features are therefore active) should automatically become much less probable than those written in Cyrillic. We would not directly obtain this sort of generalization without label-dependent features.

Namespaces and Quadratic Feature Expansion

In the previous example, we repeated each character feature twice (e.g., "ru_c" and "cyr_c") in order to produce all combinations with the target-side features ("ru", "cyr"). This is somewhat wasteful and it also makes the feature specification less readable. VW therefore implements a feature pre-processing trick.

Features can be divided into namespaces and the user can specify that features from some given namespaces should be combined automatically. This combination produces the full Cartesian product: every feature from namespace A is combined with every feature from namespace B. (We obtain a "quadratic" number of features.)

In our work, we always define two namespaces and request their combination. We call the first namespace S (source or shared) and we put all features which do not depend on the predicted class in this namespace. The second namespace is called T (target) and contains all features of the currently predicted class.

Our language identification example would be simplified as follows (the syntax which we use here is quite similar to the VW format):

```
shared |s c п а с и б о
0 |t en lat
0 |t de lat
1 |t ru cyr
0 |t uk cyr
0 |t bg cyr
```

When we request the quadratic feature expansion,[6] the final feature set is the union of S, T, and $S \times T$.

Normalization of Model Predictions

In order to obtain a conditional probability distribution $P(y|x)$ from the raw model predictions, we need to normalize the K independent scores produced by VW at test time.

The calculation of normalization depends on the loss function used in training. Note that by shifting the actual "model" to the definition of the loss function, the raw outputs of VW are very different for each of the loss functions.

Specifically, VW trained with squared loss (along with the reduction to K regressors) produces numbers *roughly* in the interval $(0, 1)$. In this case, we clip the outputs to the interval $(0, 1)$ and then divide them by the sum of their clipped values (L^1 norm):

$$P(\hat{y}|x) = \frac{\text{clip}_{[0,1]}(\boldsymbol{w}\,\text{fv}(x, \hat{y}))}{\sum_{y'=1}^{K} \text{clip}_{[0,1]}(\boldsymbol{w}\,\text{fv}(x, y'))} \tag{4.10}$$

On the other hand, when using logistic loss, the output additionally goes through a logarithm function during training; VW therefore produces scores with larger magnitudes, often negative. The suitable normalization in this case is the softmax function, see Section 4.1.

Efficient Training

VW is very fast in a single-core setting but when the training data is very large (as in our case), distributed processing is advantageous. VW supports parallelization during training through its implementation of AllReduce (Agarwal et al., 2011).

In this scheme, there are many worker nodes arranged in a tree and a single master node at the root of the tree. Workers process chunks of training data independently, synchronize their learned weight vectors with others and communicate updates through the tree.

In practice, using AllReduce yields nearly linear speed-up (i.e., using 10 nodes makes training roughly 10 times faster) which allows us to scale to large datasets.

[6] The exact command-line option in this case is "`-q st`".

5

Discriminative Models of Translation

In this chapter, we introduce and formally define our main contribution, the discriminative phrasal translation model (DPTM). The chapter is structured as follows: we first provide a mathematical definition of the model. We then describe the feature templates which we use. We explain how translation examples are extracted and how we train the model. Finally, we introduce our attempt to approximate discriminative models by a simple measure of context similarity in Section 5.5.

Recall the motivating examples of erroneous translations produced by the PBMT component of Chimera in Section 3.3. We showed that the errors have two main causes:

- inherent limitations of standard PBMT model components,
- data sparsity in the phrase table and the LM.

We propose a discriminative model which operates on the level of phrase pairs. It predicts the translation probability of phrasal translations *in the current context*. The context comprises the full source sentence and several target-side words preceding the current phrase (similarly to a LM, our model is evaluated as the translation is constructed). We use a rich linguistically motivated feature set to describe the context and to avoid data sparsity issues.

The scores predicted by DPTM are used as a single additional feature in the log-linear model of a phrase-based MT system.

5.1 Mathematical Formulation

As stated in Section 4.1, discriminative models estimate the conditional probability distribution $P(Y|X)$ where Y are labels/classes and X are inputs. In the case of MT, we assume that our inputs are sentences in the source language, $F = (\boldsymbol{f}^{(1)}, \ldots, \boldsymbol{f}^{(N)})$ and "labels" are their translations in the target language $E = (\boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(N)})$. The variables F and E denote the whole training corpus whereas $\boldsymbol{f}^{(i)}$ and $\boldsymbol{e}^{(i)}$ correspond to individual sentences.

As we discussed in Section 2.5, PBMT models predict not only the translation but also the phrasal segmentation and reordering. The mapping between source and target phrases also includes word alignment information. Also note that possible translations of the same input sentence can differ both in the number of phrase pairs and in the number of produced target words. Loosely following Zens (2008), we introduced a latent variable S (for "segmentation") that captures this information and that is predicted along with the translation.

Our model is proposed within the paradigm of PBMT. To simplify its description, we make the assumption that this additional latent variable S is also predicted by the model. However, we do not model the probability of this latent variable explicitly; we disregard $P(S)$ which corresponds to treating it simply as a uniform distribution. Finally, we make a standard independence assumption and decompose the probability $P(E|F)$ of our corpus into a product of probabilities of the individual sentences $\boldsymbol{e}^{(i)}, \boldsymbol{f}^{(i)}$. Given these assumptions, we can write:

$$P(E, S|F) = \prod_i P(\boldsymbol{e}^{(i)}, S_i|\boldsymbol{f}^{(i)}) \tag{5.1}$$

$$= \prod_i P(\boldsymbol{e}^{(i)}|\boldsymbol{f}^{(i)}, S_i) \cdot P(S_i) \tag{5.2}$$

$$\propto \prod_i P(\boldsymbol{e}^{(i)}|\boldsymbol{f}^{(i)}, S_i) \tag{5.3}$$

Equation 5.2 simply rewrites the probability distribution using the Bayes' rule. In Equation 5.3, we use the assumption that $P(S_i)$ is uniform: each term is multiplied by the same constant $P(S_i)$ and can therefore be disregarded (but we lose equality between 5.2 and 5.3).

Based on this discussion, our model only needs to learn to estimate the conditional probability distribution $P(\boldsymbol{e}^{(i)}|\boldsymbol{f}^{(i)}, S_i)$. I.e., the segmentation and reordering are not predicted but instead are treated as given in our model. This is consistent with the definition of log-linear model features – as discussed in

Section 2.5, feature scores are conditioned also on segmentation and reordering.

Let us define some additional notation: we denote source-side phrases in the i-th sentence (given a particular phrasal segmentation) $(\bar{f}_1^{(i)}, \ldots, \bar{f}_m^{(i)})$ and the individual words $(f_1^{(i)}, \ldots, f_n^{(i)})$. We use a similar notation for target-side words and phrases. When sentence indices are not needed, we will sometimes omit them and denote (any pair of) sentences simply as $\boldsymbol{e}, \boldsymbol{f}$. Similarly, we will sometimes write $(\bar{f}_1, \ldots, \bar{f}_m)$ for phrases and $(f_1, \ldots, f_n)$ for words.

The question arises how to further decompose the sentence-level conditional probabilities $P(\boldsymbol{e}|\boldsymbol{f}, S)$. DPTM decomposes them into phrase-level probabilities as follows:

$$P(\boldsymbol{e}|\boldsymbol{f}, S) = \prod_{(\bar{e}_i, \bar{f}_i)} P(\bar{e}_i|\bar{f}_i, \boldsymbol{f}, e_{start-1}, e_{start-2}, \ldots, e_{start-t}) \tag{5.4}$$

The model predicts the conditional probability of each target-side phrase based on the corresponding source-side phrase, the source-side context information (the full source sentence $\boldsymbol{f}$) and some limited *left-hand side* target context. For simplicity, we denote target words to the left of the current phrase as $e_{start-1}, e_{start-2}, \ldots, e_{start-t}$, where t determines the number of considered target words.

Because we condition on the source phrase, DPTM has similarities to the phrase table. On the other hand, by conditioning on the several preceding target words, DPTM resembles a LM as well. The information that we consider when predicting the target phrase probability is very similar to Devlin et al. (2014).

Now we describe how the phrasal probability is calculated. We define a function GEN$(\bar{f})$ which represents the possible translations of a source phrase according to the phrase table. Similarly to Section 4.1, we use $\boldsymbol{w}$ to denote the learned weight vector and the function fv$()$ to represent the vector of features. The probability under our model is then defined as follows:

$$P(\bar{e}_i|\bar{f}_i, \boldsymbol{f}, e_{start-1}, \ldots, e_{start-t}) = \frac{\exp(\boldsymbol{w} \cdot \text{fv}(\bar{e}_i, \bar{f}_i, \boldsymbol{f}, e_{start-1}, \ldots, e_{start-t}))}{\sum_{\bar{e}' \in \text{GEN}(\bar{f}_i)} \exp(\boldsymbol{w} \cdot \text{fv}(\bar{e}', \bar{f}_i, \boldsymbol{f}, e_{start-1}, \ldots, e_{start-t}))} \tag{5.5}$$

It is simply a linear model with a single vector of weights $\boldsymbol{w}$. Its score is transformed to a conditional probability distribution using the softmax function.

Note that by using the GEN($\bar{f}$) function, we formally assign a zero probability to all translations of $\bar{f}$ which are unknown to the phrase table. We describe this case only for completeness – in practice, DPTM is only evaluated on translations proposed by the phrase table.

Objective Function

As we use VW in our work, the discussion on its training criteria in Section 4.2 applies here directly. We experimented with using squared loss and logistic loss in training. Because the objective function impacts the range of the model scores, we need to consider it when normalizing model predictions. We use softmax with logistic loss as described above, for squared loss we evaluated softmax and the "clipping" normalization as defined in Equation 4.10.

Specifically, when DPTM is trained with logistic loss as the objective, model predictions are optimized to approach zero or one (the true label) *after* being transformed by the logistic sigmoid function. Raw model predictions are logits and therefore have a much higher range of values and they form nice probability distributions after the softmax transformation in our multiclass setting.

On the other hand, when training regression with squared loss, raw model predictions are optimized directly towards the true label (0 or 1). It is not entirely clear how to transform individual outputs of regression into a probability distribution for multiclass classification.

Empirically, we found that the raw predictions of the regression model often lie outside the (0, 1) interval. When we use the "clipping" normalization, many predictions are transformed into zeros. Note that in the log-linear model, we further take the log of the probability, which theoretically approaches $-\infty$ in this situation (in practice, it is limited by a large negative number) and therefore disproportionately penalizes translations which are considered incorrect by DPTM. We speculate that the MT system may be forced to accept all the suggestions by DPTM and may be unable to consider alternatives (alternatively, log-linear model weight optimization might learn to ignore DPTM entirely).

We also experimented with using softmax for normalizing the *regression* outputs (trained using the squared loss objective). However, this transformation produces much less informative probability distributions (closer to uniform, higher entropy) than those produced by the classification setting (normalized using softmax).

We carried out extensive experiments with the squared loss objective. We found no difference in intrinsic prediction accuracy of the model. However,

we failed to improve the final MT quality by adding DPTM when using this criterion.

We were originally interested in using squared loss specifically when considering smooth penalties for negative examples (see Section 5.3). Our motivation was that a regression objective might be better suited for faithfully learning the mapping between training examples and their associated loss (i.e., penalty). However, when experimenting with the BLEU penalty and squared loss, we find that the model produces even more flat, uninformative distributions. Overall, we did not find a suitable configuration or transformation of the outputs of squared loss for use in the MT log-linear model.

When optimizing the model with logistic loss, the predictions form nicer probability distributions (as explained above) which can be more naturally integrated in the MT system. We therefore use the logistic loss in all experiments reported in this work.

Global Model

In our setting, different target-side phrases are the classes that we want to predict. However, the number of possible labels is extremely large (there are millions of target phrase types in larger experiments). While the GEN function used in normalization alleviates the need to evaluate all existing target phrases for each prediction (so model evaluation is tractable), there is a problem of data sparsity in training.

Recall that in multi-class classification, an independent weight vector is associated with each possible label. Training DPTM in this way would be equivalent to building millions of different classifiers. Most of these models would have little training data (positive instances) because most phrases are rare – the distribution of phrase types follows a power law similar to Zipf's law for word types.

Additionally, many phrases share words or even subphrases and with this approach, they would be treated as distinct classes; the model would not have a chance to take an advantage of this fact.

We therefore use namespaces, label-dependent features and quadratic feature expansions in VW, as described in Section 4.2. In this setting, classes are not defined by a single label but instead by a set of features. It is up to the user to define the features associated with each class. These features are then combined with the *label-independent* features to form the final feature set.

Label-dependent features may be shared among classes. For instance, English phrases "see a cat" and "feed a cat" can share the features for words "a"

and "cat". This parameter sharing allows the model to learn e.g. that the English word "cat" usually translates to the Czech word "kočka", regardless of the current phrase or other words inside the phrase.

From now on, we assume that there are two feature namespaces – S for *shared* (=label-independent) features and T for *translation* features (=label-dependent). Features in S describe the source context, the source phrase and the limited target-side context (in other words, everything that does not change with the currently predicted translation). Features in T describe the currently predicted phrase (e.g., its concatenated words as a single indicator feature, the individual words within the phrase etc.).

There is also one more technical advantage of defining our examples this way. Because the quadratic feature expansions of VW namespaces are done on the fly, we save an enormous amount of space: we define only the features in S and T and leave the large Cartesian product $S \times T$ to VW.

5.2 Feature Set

In this section, we introduce the features used in DPTM. We first describe the linguistic annotation used for each language in our experiments and then we present the feature templates based on the linguistic information.

Linguistic Resources

While our model definition is language independent, the features do use some linguistic annotation. Theoretically, the model could also be built without any linguistic processing by only considering features based on surface forms and perhaps some heuristics (e.g., using fixed-sized word prefixes as an approximation of stems/lemmas). While this is an interesting scenario for low-resource languages, we do not evaluate such settings in this work.

We focus on four languages in our work: English (the source language in most experiments), Czech, German and Romanian. We mainly evaluate the various possible model settings on English-Czech translation. We use German and Romanian primarily to verify that our findings do not depend on the language pair at hand.

English. For English (as a source language), we use the Morče tagger (Hajič et al., 2007) to obtain lemmas and tags (Penn Treebank tagset, Marcus et al. 1993). We also parse each English sentence with the MST parser (McDonald et al., 2005). Finally, we assign dependency roles (or *analytical functions*, afun[1])

[1] `https://ufal.mff.cuni.cz/pcedt2.0/en/a-layer.html#labeling`

to words based on the PCEDT annotation (Hajič et al., 2011). This annotation scheme is loosely based on the Functional Generative Description (FGD, Sgall 1967). Examples of afun include "Sb" for subjects or "Pred" for predicates. We use the Treex (Popel and Žabokrtský, 2010) toolkit as a wrapper for this entire pre-processing pipeline.

Overall, this linguistic annotation enables us to extract (on the source side) features from surface word forms, lemmas, morphological tags and some syntactic information – specifically, we use the afun attribute and the lemma of the parent.

Czech. We lemmatize and tag the Czech data using MorphoDiTa (Straková et al., 2014). The Czech positional tagset (Hajič and Vidová-Hladká, 1998) is very detailed, it fully describes the morphological properties of each word. The information includes (fine-grained) part of speech (POS), case, number, gender, tense, degree of comparison etc. There are several thousand permissible tag values and around two thousand can be observed in a corpus.

German. We use TreeTagger (Schmid, 1994) for German. Note that the German tagset is much more coarse than the Czech tagset. For instance, case information is not included in the tags, which may limit their utility for DPTM.

Romanian. We use the tagset and tagger developed by Tufis et al. (2008). We rely on the pipeline outputs as provided to us for the participation in the QT21 joint system combination submission (Peter et al., 2016).

Note that we use morphological information on the target side but not syntactic information. We can relatively easily configure an MT system to jointly predict words along with their lemmas and tags. (The phrase table then contains morphologically disambiguated words.) This technique has been shown to work in practice and in fact enable interesting target-side models which improve translation, especially into morphologically rich languages (see e.g. the discussion of morphological LMs in Section 3.1)

However, we cannot reasonably expect a phrase-based MT system to also predict the syntactic structure of the partial translation. The addition of syntactic information in the target would introduce non-local dependencies and structural requirements on the translations which cannot be captured by a standard phrase-based model and which are simply beyond the scope of this work.

Feature Templates

In this section, we describe feature templates that we have implemented for DPTM. Note that we evaluate some of them only briefly in targeted experi-

ments. We describe the feature configurations used in the main experiments at the end of the section.

As discussed in Section 2.7, we slightly abuse the paradigm of factored MT in our work. We use factors on both the source and the target side to store linguistic information about individual words. Specifically, our factored scheme (for most experiments) is as follows:

- Source: stc+lemma+tag+parent-lemma+afun
- Target: stc+lemma+tag

The factor stc refers to supervised truecasing, see Section 3.1. Most of the feature templates that we describe here can work with any single factor and even with combinations of factors.

The following is a list of source-side feature templates. These features depend only on the input sentence:

- **Source indicator.** A single feature; the concatenation of all words inside the current source phrase.
- **Source internal.** Each word inside the current source phrase represented as a single feature. Word position within the phrase is ignored.
- **Source context.** Words in a fixed-sized window around the current source phrase represented as separate features. Word position relative to the current phrase is included in the feature string.
- **Source sentence bag of words.** All words in the current input *sentence* (including the current source phrase) represented as individual features. Word position within the sentence is ignored.
- **Source sentence bigrams.** All bigrams (pairs of consecutive words) in the current input sentence (including the current source phrase) represented as individual features. Word position within the sentence is ignored.

In the next list, we define features which require target-side context information. Their depend also on the current partial translation (words to the left of the currently predicted target phrase).

- **Target context.** Words in a fixed-sized window *before* the current target phrase represented as separate features. Word position relative to the current phrase is included in the feature string.

- **Bilingual context.** Pairs of source-target words. For each word in a fixed-sized window before the current target phrase, we extract the corresponding source words using word alignment information. We generate a separate feature for each word pair found. The relative position of the target context word is also included. Bilingual context features are directly inspired by bilingual LMs (Niehues et al., 2011).

- **Target context bigrams.** All bigrams in a fixed-sized window before the current target phrase, along with their relative position.

Finally, we define the *translation* features which describe the currently predicted target phrase. To connect this division of features with the discussion on feature namespaces above: all features presented until now are part of the *shared* namespace S, they remain constant when we change the translation of the current phrase. The features that we will introduce now are in the *translation* namespace T – they describe the currently predicted class. (In other words, they are label-dependent.)

- **Target indicator.** The concatenation of all words inside the current target phrase.

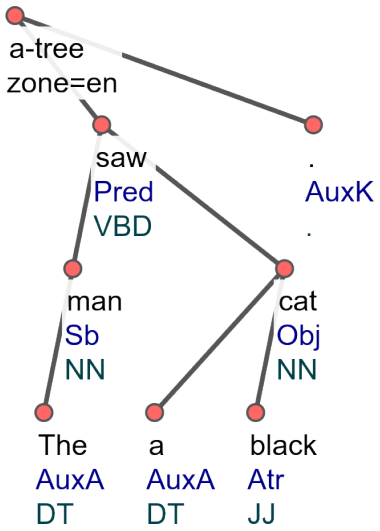
- **Target internal.** Each word inside the current target phrase represented as a single feature. Word position within the phrase is ignored.

- **Target bigrams.** All bigrams in the current target phrase represented as individual features; their position is ignored.

- **Translation scores.** Phrase table feature scores represented as dense features with real-valued weights. (All other features are sparse indicators with weight 1.0).

Figure 5.1 illustrates our feature set on a sample sentence. We show the source-side dependency tree, the factored representation of the input sentence and of the partial translation. For each feature template category (e.g. source indicator), we show only a handful of features and we try to vary their factor configurations as well to illustrate the flexibility.

Note that as stated before, the final feature set is in fact the union of S, T and the full Cartesian product $S \times T$. This expansion generates features such as "stc:a_black-lemma:černý" or "stc-stc-1:saw:uviděl-stc:černou" which describe the translation in its context and give the model some generalization power. For instance, DPTM can learn that "black" tends to translate as "černý"



Dependency parse of the input sentence.

stc	the	man	saw	a	black	cat	.
lemma	the	man	see	a	black	cat	.
tag	DT	NN	VBD	DT	JJ	NN	.
afun	AuxA	Sb	Pred	AuxA	Atr	Obj	AuxK
plemma	man	saw	<root>	cat	cat	saw	<root>

Factored representation of the input sentence, current phrase highlighted.

stc	muž (the man)	uviděl (saw)	černou *(a black)*
lemma	muž	uvidět	černý
tag	NNMS1——A—-	VpYS—XR-AA—	AAFS4——-1A—-

Partial translation, current phrase highlighted.

Source indicator	stc:a_black tag:DT_JJ afun:AuxA_Atr . . .
Source internal	lemma:a lemma:black stc+tag:a_DT stc+tag:black_JJ . . .
Source context	lemma-3:the lemma-2:man lemma-1:see lemma+1:cat . . .
Source BOW	lemma:the stc:the lemma:man stc:man . . .
Source bigrams	stc:the_man stc:man_saw . . .
Target context	stc-1:uviděl tag-1:VpYS—XR-AA— stc-2:muž . . .
Bilingual context	stc-stc-1:saw:uviděl stc-lemma-1:saw:uvidět . . .
Target context bigrams	stc-2:muž_uviděl
Target indicator	stc:černou lemma:černý tag:AAFS4——-1A—- . . .
Target internal	lemma+tag:černý_AAFS4——-1A—- . . .
Target bigrams	—
Translation scores	p-direct:0.1 lex-direct:0.013 p-inverse:0.5 lex-inverse:0.007

Extracted features.

Figure 5.1: Example of feature extraction.

Feature Type	Configurations	
	Czech A	Czech B
Source Indicator	s, l, l+t, t	l, t
Source Internal	s, s+a, s+p, l, l+t, t, a+p	l, l+a, l+p, t, a+p
Source Context	s (-3,3), l (-3,3), t (-5,5)	l (-3,3), t (-5,5)
Target Context	s (2), l (2), t (2), l+t (2)	l (2), t (2)
Bilingual Context	—	l+t/l+t (2)
Target Indicator	s, l, t	l, t
Target Internal	s, l, l+t, t	l, t

Table 5.1: Feature templates used for English-Czech translation. Letter abbreviations refer to word factors: s (stc), l (lemma), t (morphological tag), a (analytical function), p (lemma of dependency parent). Numbers in parentheses indicate context size.

from all phrases which contain these words, and thanks to lemmatization, it can learn that this correspondence holds regardless of the Czech inflected form ("černá", "černou", "černými" etc.).

From other examples, it can learn that nouns tend to be accusative when preceded by adjectives in accusative, or that subjects (afun "Sb") tend to translate into nominative case.

Feature Configurations

Table 5.1 specifies the basic feature configurations used in our English-Czech experiments. Originally, we worked with the first variant (Czech A) in the table but later found that the second configuration tends to perform better.

Some types of features are not listed in Table 5.1 at all. Specifically, we do not use translation model scores, bag-of-words features or any of the bigram features in our experiments.

In preliminary evaluations, all bigram feature types worsened model accuracy as well as the final MT quality. We do not cover these experiments in our work. Possibly, these features could prove useful if the learning parameters and regularization were carefully tuned.

We conduct a separate set of experiments with the translation model features, see Section 7.3. In our main experiments, we opt not to use this feature type.

We evaluated bag-of-words features and found that they increase the training set size considerably and do not bring a significant improvement. However, more experimentation would be required to draw reliable conclusions.

5.3 Extraction of Training Examples

Before we train DPTM, we extract training examples. Our extraction procedure requires the parallel data and a phrase table trained on this data (which we use to look up alternative translations of phrases). We go over each sentence in the parallel corpus. We assume that both the source and the target sentences contain (as factors) the linguistic information described above.

For each phrase pair which can be extracted from the sentence pair, we generate a set of training examples. We look up all possible translations of the current source phrase in the phrase table. The current target phrase is a positive example and all other phrasal translations serve as negative examples.[2]

Because phrase tables tend to be very large and moreover, most of their content is redundant and/or noisy, we perform significance test filtering of the phrase table as proposed by Johnson et al.. The filtering reduces the phrase table to a fraction of the original size and allows us to greatly speed up the extraction of training examples. Because there are fewer training examples and fewer possible translations per phrase, we also save a large amount of disk space when generating the feature extract files.

Loss of Negative Examples

With the feature file format that we use, we can define the loss associated with selecting a wrong translation separately for each negative example. Our basic setting is 0/1 loss: when the system selects the correct translation, it pays no penalty. When any other translation is chosen, it pays a penalty of one.[3]

As an alternative, we follow the suggestion made by Marine Carpuat at the JHU Workshop on domain adaptation in MT (Carpuat et al., 2012) and we base the loss on BLEU, which makes the penalty for negative examples more smooth. The motivation is that translation candidates which only differ very slightly from the correct target phrase should perhaps not be penalized as heavily as completely different, unrelated translations. Note that in phrase-

[2] We could probably subsample negative examples for improved efficiency but we have not experimented with such settings.

[3] Note that so far, we have used 1 to denote correct labels (positive examples), as is customary in ML literature.

based systems, phrasal translations can often only differ in minor details such as punctuation.

Because BLEU is a document-level metric, its application to individual sentences, let alone phrases, is problematic. Recall that BLEU is defined as follows:

$$\text{BLEU} = \text{BP} \cdot \exp \sum_{i=1}^{n} (\lambda_i \log p_i) \tag{5.6}$$

BP stands for brevity penalty which discourages translations shorter than the reference. The terms p_i in the equation are n-gram precisions for n-gram size 1 up to 4. Our phrases will rarely contain four or more words. Even more rarely, any 4-gram will match with the correct translation. BLEU is undefined (or zero, as is often done in practice) in these cases.

There are several smoothing techniques which address this problem. We opt for using BLEU+1 (Lin and Och, 2004) where for each n-gram size we add a single *virtual* matching n-gram to avoid zero precision. Moreover, because target phrases are mostly short, we only look at n-grams up to size two; our loss could be formulated as "1 - BLEU2+1".

Translation	**1 - BLEU2+1**
sušeného odstředěného mléka	0
odstředěného mléka ,	0.29
odstředěného mléka na	0.29
odstředěného mléka se	0.29
odstředěného sušeného mléka	0.42
odstředěného mléka	0.63
odstředěné mléko vyrobené	0.71
sušeným odstředěným mlékem	0.71
se sušeným odstředěným mlékem	0.78
odstředěné mléko	0.85
odstředěné mléko	0.85
odstředěné	0.90

Figure 5.2: BLEU loss for negative examples: alternative translations of the English phrase "skimmed milk". The correct translation has a loss of zero.

Figure 5.2 shows an example of our BLEU loss applied in the model training data. The correct translation (in the current context) is "sušeného odstředěného mléka" and therefore has a loss of zero. Translation candidates which only differ slightly from this translation obtain a smaller loss. During training, the model will therefore be penalized less for producing similar translations.

Leaving One Out

The training data for DPTM and the phrase table is identical. This can potentially lead to model overfitting: when we use translation model probabilities as features, our model will learn to rely too heavily on their values. To counter this effect, we use leaving one out in a way similar to Wuebker et al. (2010).

In machine learning, leaving one out is a form of cross-validation where the model is always estimated on all available data except for a single example. The model is then evaluated on this "left-out" example. This procedure is typically expensive but avoids the potential problem of standard cross-validation where unbalanced folds can lead to unreliable estimates of true model performance.

We perform leaving one out during the extraction of training examples for DPTM. Our aim is to cancel the contribution of the current phrase pair from the phrase table. We use a phrase table format which stores not only phrasal translation probabilities but also their number of occurrences in the parallel data, from which the probabilities can be calculated on the fly according to Equation 2.1.

Before we extract a training example from a given phrase pair $\bar{e}, \bar{f}$, we first remove its contribution from the raw counts by subtracting one from the counts of $\bar{e}$, $\bar{f}$ and $\bar{e} \wedge \bar{f}$. The number of co-occurrences can drop to zero in which case we do not extract a training example at all (the phrase pair was only observed in the current sentence).

If the phrase pair "survives", we recalculate its discounted direct and inverse translation probabilities. The feature extraction then uses the discounted values.

Note that because training instances can be filtered out, leaving one out affects model training even when translation model probabilities are not included in its feature set.

Efficiency

The training files with features can be very large. For example, when extracting examples from 1 million English-Czech sentence pairs, we process 22 million phrase pairs, each creating a single positive and multiple negative examples. The total size of the feature extract files (in the text format) is roughly 58 GB, which corresponds to around 5 GB after gzip compression.

In order to create such large training files (relatively) efficiently, we take advantage of the fact that features from a single sentence pair do not depend on any other sentences in the data. This allows us to trivially parallelize the task of

feature extraction: we simply distribute sentence pairs evenly among multiple computer nodes in our cluster and extract features separately. We then concatenate the outputs. Note that our round-robin distribution of sentences to workers changes the order of the training examples; this is in fact desirable for online learning (stochastic gradient descent) because it makes the order more random.

Finally, let us note that we also run feature extraction in multiple threads on each node. Together, these optimizations allow us to produce the training files relatively quickly even for large parallel corpora.

5.4 Training

We train the model using VW. For efficiency, we use the AllReduce parallelization scheme (see Section 4.2) which allows us to distribute training jobs across cluster nodes. We usually run around 10-20 jobs; the provided speed-up is roughly linear.

In order to avoid overfitting, we measure the accuracy of the model after each pass over the training data. We use a fixed held-out dataset for this evaluation. We typically run 10 passes over the data and select the model with the best accuracy afterwards.

Other parameters of VW include hash size which we set to 26 or 28 depending on the data size and feature set. We experiment with different values for regularization, see Section 7.3. As discussed before, we train the model with logistic loss. All other parameters of VW are left to their default values.

Finally, because the behavior of VW can change over time, we use our forked version of VW which is frozen and which contains minor fixes of the installation files.[4]

5.5 Context Similarity Feature

This concludes the description of our model. Now, for completeness, we report on a preliminary experiment which was done before the development of DPTM. With the goal of improving mainly lexical choice, we implemented an additional feature for the log-linear model in the decoder which scores phrases without the need for a classifier, relying on a simple measure of similarity. The motivation was to explore whether a very simple solution might also be effective for this task.

[4] `https://github.com/moses-smt/vowpal_wabbit`

Given a phrase pair $(\bar{e}, \bar{f})$, the feature computes the cosine similarity between the current context and the contexts in training data where $\bar{f}$ was translated as $\bar{e}$. The feature therefore promotes translations which occurred in a similar context. In our experiments, we worked with context size of 3 words on each side, disregarding their position.

We denote the sets of context words which we compare A and B. If we disregard word counts in the context vector, cosine similarity is reduced to:

$$\text{sim}_1(A, B) = \frac{|A \cap B|}{\sqrt{|A|} \times \sqrt{|B|}}$$

We experimented with this simplified measure and with a formula that takes word frequencies into account:

$$\text{sim}_2(A, B) = \frac{\sum_i A_i \cdot B_i}{\sqrt{\sum_i A_i^2} \times \sqrt{\sum_i B_i^2}}$$

We evaluated a range of experimental settings, however the feature was never beneficial. Even parameter tuning assigned nearly zero weight to it. We attribute the result (at least partially) to the following problems:

- The feature heavily depends on phrase segmentation.
- No abstraction from surface forms is done.
- Function words have the same weight as content words.

The feature is very unstable: if the decoder chooses a slightly different segmentation of the input, observed contexts of the individual phrases change dramatically and the feature score is very different. We could perhaps mitigate this issue if we disregard function words and punctuation and also if we use e.g. lemmas instead of word forms, reducing the sparsity of the observations.

However, a more principled solution is indeed to use a discriminative classifier with a rich set of features – issues such as the distinction of function and content words, abstraction to lemmas or sensitivity to phrase boundaries are naturally resolved by the representation of context and the utilization of machine learning.

The classifier can learn what is a suitable generalization for this task and automatically downweigh the unimportant features (such as punctuation). Also, as it is presented with all possible phrasal segmentations during training, it can discover a robust set of features which are not sensitive to shifted phrase boundaries.

6

Integration in Phrase-Based Translation

So far, we have introduced DPTM as a stand-alone model. In this chapter, we describe the integration of DPTM into a phrase-based MT system. Our model is integrated directly in the Moses phrase-based decoder and is available in the toolkit as a standard feature function.

Recall the definition of the log-linear model in Section 2.5. The score of a (partial) translation produced by the system is calculated using this equation. We integrate our score by adding another *feature* into the log-linear model.[1]

6.1 Motivation

In this section, we describe why it is desirable to integrate our model in the translation process as tightly as possible.

When additional features are added to MT systems, their integration in decoding can be problematic. Imagine for instance using a dependency parser score as a feature (in order to promote syntactically well-formed translations). Such a feature is most easily evaluated when the full translation is already available. On the other hand, evaluating partial translations is difficult.

In such situations, n-best list rescoring can be used instead. The decoder translates the input without the additional feature and produces n best translations of each input sentence. The model then evaluates only the full transla-

[1] To avoid confusion, we will only use the term feature to refer to the log-linear model features within this chapter.

tions and possibly causes a different translation to be selected as the top candidate for further evaluation.

The main disadvantage of such solutions is the limited impact of the model – many possible translations are pruned during search. In fact, n-best lists do not really represent the search space well. Most alternative translations in an n-best list are typically very similar to each other (Gimpel et al., 2013). The model therefore has a very limited set of translations to choose from. When integrated in the search directly, the model can guide the MT system as the translation is constructed and reach translations that would otherwise not appear in the n-best list.

Because the evaluation of DPTM (esp. when target-context information is considered) is expensive, we first experimented with n-best list rescoring as well. We obtained modest gains when using our model but the improvements were limited due to the reasons discussed above. We do not report on these experiments in this work and instead describe a relatively efficient way of using DPTM directly in the search.

6.2 Evaluation with Source Context

When target-context information is not used, the integration of DPTM is relatively simple. Recall the description of decoding in Section 2.6. The first step when a new input sentence arrives is to collect translation options for all source spans from the phrase table.

After this step, we already have all the information required for evaluating our model: the source context for each span is constant and we have all the possible translations at hand (so that we can calculate the normalized conditional probability).

When target context is disabled, we therefore fully evaluate our model even before decoding starts (but after translation options are collected) and add the model score to each possible translation of each source span. Note that we need to retrieve all translations for a source span in a single step to make normalization efficient.

According to our description of decoding in Section 2.6, our model is a *stateless* feature when target-side context information is not used.

When we evaluate a set of span translations, we first extract the relevant features for the source context. These features are shared across the possible translations of the phrase (they belong to the namespace S as defined above). We make use of VW feature hashing to store the features efficiently.

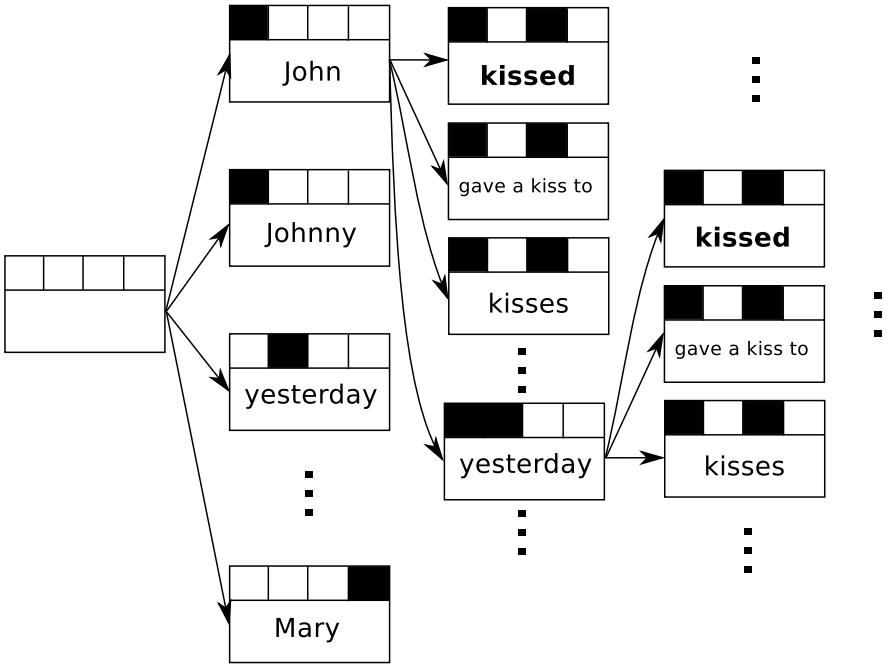


Figure 6.1: Multiple evaluations of a single translation option during decoding.

We then go over all the possible translations. For each target phrase, we generate its features (in the namespace T), add them to the object holding the source-context features and we evaluate our model (with quadratic feature expansions generated automatically on the fly). We remove the namespace T again and move on to the next translation. When we have evaluated all of the translations of the input span, we normalize the scores.

The most computationally expensive part in this scenario is feature extraction. It is a sequence of many relatively expensive string operations. However, because a source-context-only model is evaluated only once for each span translation, the cost is negligible compared to the decoding time.

6.3 Evaluation with Target Context

Recall the discussion about LMs in decoding. The LM is a *stateful* feature because it is non-local: the LM probability of a phrasal translation depends on the target-side words *preceding* this translation.

When we add target-context features to DPTM, the same argument applies. Our model now needs to be evaluated many times for each target translation – every time it appears in a new target-side context. Figure 6.1 shows an exam-

ple: the target phrase "kissed" follows the words "<s> John" the first time and the words "<s> John yesterday" the second time. This is just a subgraph, this phrase appears many times during a full search. Also note that because DPTM score is locally normalized, we need to evaluate all the possible translations of the source phrase ("políbil", see Figure 2.2) before producing the probability of "kissed" in each context. For our feature, the hypothesis *state* is defined as the last t words (for target context size t).

The most naive approach to integrating the model would be to do the following operation every time a partial hypothesis is expanded with a phrasal translation $(\bar{e}_i, \bar{f}_i)$:

1. Generate the source-side and target-side context features.
2. Fetch all possible translations $\bar{e'}_i$ of the current input span $\bar{f}_i$.
3. Generate features and evaluate the model for each translation $\bar{e'}_i$.
4. Normalize the predictions and return the probability of the proposed translation $\bar{e}_i$.

This naive procedure is very computationally expensive and makes experimenting on a large scale infeasible. Also, DPTM could not be realistically used in practice with this approach. We therefore make several optimizations which we describe in the following sections.

Separation of Source-Context and Target-Context Evaluation

One advantage of having a simple linear model is the possibility of dividing its evaluation into multiple stages. We can categorize the features in the feature vector as follows:

1. Source-context features (namespace S), denoted f_{src}
2. Target-context features (namespace S), denoted f_{tgt}
3. Translation features (namespace T), denoted f_t

Then the (unnormalized – raw) score of our model is defined as:[2]

$$s = \boldsymbol{w} \cdot \{\underbrace{f_{src} \cup f_{tgt}}_{S} \cup \underbrace{f_t}_{T} \cup \underbrace{\{\{f_{src} \cup f_{tgt}\} \times f_t\}}_{S \times T}\} \tag{6.1}$$

[2]We ask the reader to excuse the rather creative notation which mixes sets and vectors. Hopefully, the meaning is apparent from the equations.

The formula is the dot product between the weight vector and the feature vector, which we divide into categories as described above. Note that $S \times T$ corresponds to the quadratic expansion of the two namespaces. This expansion is carried out automatically inside VW.

Thanks to distributivity of multiplication, we can split the dot product into several parts (calculated at different stages) and add them together later to arrive at the final score.

All features except for f_{tgt} are available before decoding which allows us to calculate the "source-context only" part of the score ahead of time. Also note that we need to calculate this part only once for each target phrase. The "source-context only" part is calculated as follows:

$$s_{src} = \boldsymbol{w} \cdot \{f_{src} \cup f_t \cup \{f_{src} \times f_t\}\} \tag{6.2}$$

We store the raw s_{src} for each phrase pair in cache and combine it with the rest of the score afterwards. During decoding, when we need to evaluate the translation in its target context, we compute the "target-context only" part of the score:

$$s_{tgt} = \boldsymbol{w} \cdot \{f_{tgt} \cup f_t \cup \{f_{tgt} \times f_t\}\} \tag{6.3}$$

Note that we include the translation features f_t in both equations. This is a technical necessity because we rely on VW for the quadratic feature expansions. This means that the contribution of f_t is counted twice. To obtain the correct result, we therefore need to subtract the contribution of f_t again. The final model score is calculated as follows:

$$s = s_{src} + s_{tgt} - \boldsymbol{w} \cdot f_t \tag{6.4}$$

Note that in terms of size, both f_t and f_{tgt} are usually quite small compared to f_{src}. This trick allows us to do most of the expensive feature generation only once before decoding.

Caching of Feature Hashes

Generating feature values from string representations is an expensive operation which we would like to avoid as much as possible. VW internally represents features by their hashes. This allows us, once we generate a feature vector, to obtain the hashes from VW and store the generated feature identifiers simply as a vector of integers (along with their floating-point values) for future use.

In the previous section, we described how the partial (source-context) score can be calculated. During this optimization, we generate the features f_t for each translation option in the current sentence. It would be wasteful to create these features again so we cache the generated feature vector associated with each translation option at this time. During decoding, we simply look it up in cache.

We have successfully decreased the number of operations required when evaluating a hypothesis expansion. With all the optimizations described so far, DPTM evaluation of a hypothesis expansion looks roughly as follows (if we disregard normalization for a moment):

1. Look up the source-context only partial score s_{src}.
2. Look up the generated features f_t for the current translation.
3. Generate target-context features f_{tgt}, calculate s_{tgt} and finally s.

Again, we would like to avoid generating features. We note that target-side contexts can often repeat during the decoding of a single sentence. We therefore also cache f_{tgt}. Whenever we evaluate a new hypothesis, we first check whether we have seen its current context and if so, we simply load its feature vector from cache.

With these optimizations, we often avoid generating features entirely. Hypothesis evaluation then amounts to loading a single number from cache (the partial score), the computation of the dot product between the (quite small) feature vector and the weight vector $\boldsymbol{w}$ (which VW handles very efficiently) and several addition operations.

While still not comparable to the handful of operations done by optimized LMs (e.g. KenLM, Heafield 2011), these tricks make using DPTM for large-scale experiments viable.

Caching of Final Results

We have already stated several times that the model score needs to be locally normalized. When scoring a translation $\bar{e}_i$ in a given context, we also need to evaluate all the alternatives $\bar{e'}_i$. Once we have calculated these scores, there is no reason to throw them away, we therefore store them in a cache as well.

Because the phrase-based decoder will probably try most of the alternative translations anyway, we can save a lot of computation.

When we combine all the three tricks described above, we arrive at the complete algorithm illustrated in Figure 6.2.

```
function EVALUATE(t, s)
    span = t.getSourceSpan()
    if not resultCache.has(span, s) then
        scores = ()
        if not stateCache.has(s) then
            stateCache[s] = CtxFeatures(s)
        end if
        for all t' ← span.tOpts() do
            srcScore = srcScoreCache[t']
            c.addFeatures(stateCache[s])
            c.addFeatures(translationCache[t'])
            tgtScore = c.predict()
            scores[t'] = srcScore + tgtScore
        end for
        normalize(scores)
        resultCache[span, s] = scores
    end if
    return resultCache[span, s][t]
end function
```

Figure 6.2: Algorithm for obtaining classifier predictions during decoding. The variable t stands for the current translation, s is the current state and c is an instance of the classifier (Tamchyna et al., 2016a).

Evaluation of Decoding Speed

In order to evaluate how effective our optimization is, we added our model to a baseline English-Czech PBMT system trained on the full CzEng 1.0 corpus (Bojar et al., 2012). The training data size is around 14.8 million parallel sentences. We measure the average time required to translate a single sentence in three settings: the baseline, naive integration (includes the final trick – caching of final results) and the optimized solution. The baseline is naturally the fastest system and requires only 0.8 seconds on average. The naive implementation increases this time to 13.7 seconds. When we include all optimizations, we decrease the time to 2.9 seconds.

While the system is still several times slower than the baseline, the addition of DPTM does not prevent us from running large-scale experiments any more. In fact, we were able to thoroughly evaluate hundreds of experimental settings, which would have not been possible without the optimization.

7 Experimental Evaluation

In this chapter, we describe our experiments with DPTM. Section 7.1 introduces our baseline system for English-Czech translation. In Section 7.2, we present our primary experiments where we evaluate the benefit of DPTM for English-Czech translation in a variety of settings. We also provide a manual analysis of the results. Section 7.3 focuses on the discriminative model in more detail, evaluating different settings for model training. In Section 7.4, we present the results of applying DPTM to other languages and to translation into English. Finally, Section 7.5 describes a small annotation experiment in which we attempt to set an upper bound on prediction accuracy for target-side morphology when only source-side context information is available.

7.1 Baseline Setting

We first describe our baseline system(s) for English-Czech translation. We also evaluate the impact of significance test filtering and additional target-side factors on translation quality of the baseline.

Corpora

In our English-Czech experiments, the data that we use is largely similar to a constrained Chimera setup. We use (subsets of) CzEng 1.0 (Bojar et al., 2012) as our parallel training data. We always use the same data for training DPTM and the phrase table. We only use the target side of the parallel data for training the LM; no additional monolingual corpora are included.

We use WMT test sets from various years as our development and test corpora. In most experiments, WMT13 is our development set and we test on WMT14 test set.

We follow the data processing pipeline as described in Section 5.2.

Training Pipeline

We use standard tools for building PBMT systems bundled with the Moses toolkit. We use Eman (Bojar and Tamchyna, 2013) extensively to keep track of experiments and results.

We use standard settings for phrase table training. We set the maximum phrase length to 7 words. We do not smooth the phrasal translation probabilities. As described in Section 5.3, we use significance test filtering of the phrase table. This filtered phrase table is used both for training the DPTM and in the final MT system. We evaluate whether this filtering affects translation quality below. Because our model requires target-side morphological information, our phrase table is factored and translates from truecased surface forms (stc) into stc+lemma+tag.

Unless specified otherwise, we use a single 5-gram LM over surface forms (stc) with modified Kneser-Ney smoothing (Chen and Goodman, 1999). Our implementation of choice is KenLM (Heafield, 2011) with the default parameters.

We use minimum error rate training (MERT, Och 2003) for optimizing the weights of the log-linear model. Due to stability concerns, we typically run MERT five times and report the average. We use MultEval (Clark et al., 2011) to determine whether the differences between systems are statistically significant.

Effect of Phrase Table Filtering

Filtering of the phrase table helps DPTM scale to large data – if we used the original phrase table, the number of phrases and possible translation options for each source phrase would make feature extraction even more computationally expensive and the training files might become unmanageably large. By removing singleton phrases and rare/noisy phrase pairs, phrase table filtering possibly also helps DPTM avoid overfitting during training.

Because filtering is necessary for DPTM, we need to verify that it does not hurt translation quality. Otherwise, it might happen that our model simply

mitigates the loss caused by phrase table filtering and would not improve translation quality (as much) over a baseline with an unfiltered phrase table.

Johnson et al. report no decrease and sometimes even improvements of translation quality when applying significance test filtering. We still need to verify that this holds for our setting as well.

We therefore ran a small experiment where we compare two settings:

- A baseline system without significance test filtering.
- A baseline system with the filtered phrase table.

We use 1M parallel training sentences from CzEng 1.0 for training the system. As in our other experiments, we run MERT five times and use MultEval to compare the results.

System	Table Size	BLEU (WMT14)
baseline	29.7M	13.2
+filter	4.7M	13.2

Table 7.1: Effect of phrase table filtering.

Table 7.1 shows the obtained results. Phrase table size is reduced to roughly 16% of the original size but the average BLEU remains exactly the same. This result shows that we do not weaken our baseline system by using phrase table filtering.

Additional Target-Side Factors

Our feature set requires that the target side contains not only surface word forms but also lemmas and morphological tags. When a phrase table is trained this way, we make its training data somewhat more sparse: instead of simply obtaining statistics for each surface form, we count the concatenations of form+lemma+tag and estimate translation probabilities from them.

To illustrate the problem, consider the Czech word "hrad" ("castle"). Its nominative and accusative cases are identical. The following sample parallel corpus shows that the translation is the same both when "hrad" is the subject or the direct object in the sentence:

The **castle** stands on the hill.	**Hrad** stojí na kopci.
The army conquered a **castle**.	Armáda dobyla **hrad**.

When we "disambiguate" the form "hrad" by appending its morphological tag to it, we force the statistical model to count the statistics of the two words separately, which introduces data sparsity.

To quantify this effect, we count the number of distinct Czech word forms in the CzEng 1.0 parallel corpus and compare it to the number of distinct form+tag combinations. There are around 1.29 million distinct word forms and roughly 2.17 million distinct form+tag concatenations in the corpus, which is quite a prominent difference.

In Chimera, we use these sparse models because the benefit of LMs over target-side morphological tags clearly outweighs the loss. In this set of experiments, our aim is to verify that we do not lose more by the data sparsity than we gain by applying DPTM. Note that as described above, we only use a single LM over surface forms in most of our experiments; the additional target-side factors are therefore only utilized by DPTM. (Experiments which include these additional LMs are described later in Section 7.2.)

We evaluate the difference in BLEU for two sizes of training data, 1 million and 5 million parallel sentences. We train a pair of baseline systems for each data size. The first system only produces surface word forms while the other system also outputs lemmas and tags. Note that BLEU is always measured over word forms only.

	stc	stc+lemma+tag
1M	**13.2**	13.0
5M	**15.5**	15.2

Table 7.2: Effect of using additional target side factors in a baseline system.

The results in Table 7.2 show that there is a small but statistically significant decrease of BLEU for both data sizes. However, the improvements gained by using DPTM outweigh this loss.

7.2 Discriminative Models for English-Czech Translation

This section contains our main experiments. We evaluate the utility of DPTM for English-Czech translation under various conditions. Some of these results were presented in Tamchyna et al. (2016a).

Effect of Training Data Size

The goal of this set of experiments is to answer the following two questions:

- Does the discriminative model scale to large data sizes?
- Is target-side context information useful?

We train systems on subsets of CzEng 1.0 of the following sizes:

- small – 200 thousand sentence pairs,
- medium – 5 million sentence pairs,
- full – roughly 14.8 million sentence pairs.

We use (subsets of) the feature set A (see Section 5.2) in these experiments. We compare a baseline system, a system with source-context features only and finally, a system with a full feature set including target-context information.

	small	medium	full
baseline	10.7	15.2	16.7
+source	10.7	**16.0**	**17.3**
+target	**11.2**	**16.4**	**17.5**

Table 7.3: BLEU scores obtained on the WMT14 test set (Tamchyna et al., 2016a).

Table 7.3 shows the obtained BLEU scores. Statistically significant differences (α=0.01) are marked in bold. When evaluating statistical significance, we compare the "+source" systems to the baselines and the "+target" systems to "+source".

For the smallest setting, DPTM with source-context features only brings no improvement. However, the addition of target-side context on top of the source-context only model improves the BLEU score by 0.5.

When we move to the medium-sized setting, source-context features become effective and improve BLEU by 0.8. Target-side context brings an additional significant improvement of 0.4 points.

Finally, in the full-data setting, the differences remain significant, albeit somewhat smaller: source-context features add 0.6 BLEU and target-context features add another 0.2 BLEU on top of the baseline.

This is an encouraging finding. It shows that with the optimizations, we are indeed able to scale up to realistic data sizes. More importantly, target-side context information seems useful as it consistently improves translation quality.

Target Context Size

In our original preliminary experiments, we found that target context of size two seemed the most promising. In the following set of experiments, we evaluate different context sizes thoroughly in full translation systems.

Our setting is again English-Czech translation. We use 1M parallel sentences from CzEng 1.0 as our training data, we tune on WMT13 and evaluate on WMT14 test set. We use feature set B as the starting point and we vary the target context size from 1 to 5 preceding words. For each setting, we need to run feature extraction and model training again (because the feature set changes). We then run MERT five times and report the average, similarly to other experiments.

Here we are not only interested in the final BLEU score. We would also like to answer the following questions:

1. How does the context size affect model accuracy (intrinsic evaluation)?
2. What is the impact on decoding speed?
3. Is there any effect on the weight assigned by MERT?

Table 7.4 lists all results. Let us discuss the first question, intrinsic accuracy. We evaluate the VW model on a held-out set which comes from a rather different domain than most of its training data (news as opposed to the mix of domains in CzEng 1.0).[1] Note that target-context features in the evaluation are extracted from the *true* (gold) context, which means that the model is cheating when compared to the source-context setting only (context size 0).[2]

Nevertheless, we can make some interesting observations based on the results. There is an improvement of accuracy only when we go from a single word to two words of context. This difference of 0.6 is quite convincing but when we increase the size, held-out accuracy begins to drop. We attribute the decrease to model overfitting – target context of this size probably helps the model memorize the training data better and so the weights of context features may be set too high. It is also possible that long target context does not generalize as well across domains, making the model more sensitive to domain shift.

In terms of decoding speed, the results confirm our expectation. The baseline system (no discriminative model) is the fastest one, requiring only around 0.28 seconds per sentence. The addition of the source-context model slows the

[1] The test set which we use to measure BLEU is the news domain as well.
[2] We discuss this issue in more detail in the Section 7.3, Intrinsic Evaluation.

Context Size	Accuracy	Secs Per Sentence	BLEU
0	70.442	0.36	13.8
1	77.982	0.75	**14.0**
2	**78.584**	1.00	**14.0**
3	78.517	1.18	13.4
4	78.498	1.28	13.7
5	78.220	1.47	13.6
baseline	–	**0.28**	13.0

Table 7.4: Effect of target context size.

system down somewhat: even though we do not need to evaluate different target contexts, we still need to score each translation option in its source context, which takes a small amount of time.

When we add target context, DPTM begins to track the feature state in decoding. As described in Section 6.3, this leads to multiple model evaluations for different target contexts and prevents hypothesis recombination. As the context grows larger, we observe a steady decrease in decoding speed down to 1.47 seconds per sentence.[3] However, it is encouraging to see that a large context size does not make decoding prohibitively expensive, and that our implementation seems to scale relatively well in such a setting.

Final translation quality as measured by BLEU shows an interesting pattern: without target-side context, DPTM achieves a lower BLEU score (13.8) than when target context of a *suitable* size is used. At least in this data setting, the ideal context size appears to be either 1 or 2 preceding words. The best context size can of course depend on many factors (training data size, the domain of evaluation data etc.) and it should be ideally optimized by grid search on a held-out set. Due to the number of experiments in this work, we do not optimize this variable and instead use target context of size two (unless stated otherwise).

We also make an observation regarding MERT when context size varies. We find MERT increasingly unstable as context size increases. We perform additional 5 optimization runs for each setting giving us a total of 10. We calculate the standard deviation of the final BLEU score for each context size and

[3]Note that the times reported in this section are much lower than in Section 6.3. The reason is training data size; here we only trained the MT system on 1 million sentence pairs, resulting in a much smaller phrase table, LM and VW model.

observe a steady increase from 0.12 (context size 1) up to around 0.7 (sizes 3 and higher).

We further inspect the optimized model weights for cases where MERT was particularly unsuccessful and while it is difficult to interpret the numbers directly, we find that DPTM obtains a suspiciously high weight (around 0.2) in all of these cases. Our (very) intuitive interpretation is that by including a lot of the target context information, our model becomes a substitute for both translation model scores *and* the LM. MERT can easily improve BLEU by giving a higher weight to DPTM. However, better translation quality could be reached by a more fine tuned combination of all components (recall that our model optimizes cross-entropy, not the final evaluation metric) but this combination is harder to find.

Using different optimization algorithms such as pairwise ranking optimization (PRO, Hopkins and May 2011) or batch MIRA (Cherry and Foster, 2012) could provide further insights into this problem. Even though all approaches include some degree of randomness, these alternative algorithms are empirically much more stable than MERT. On the other hand, we found that at least in the case of PRO, the output is often suboptimal (albeit stable) and thus the results may not provide a complete picture when comparing systems close in performance (Tamchyna and Bojar, 2013).

Language Models over Morphological Tags

Our baseline setup does not include a LM over morphological tags. The goal of target-side context features in DPTM is primarily to improve morpho-syntactic coherence. Arguably, a LM over tags has a similar role; we therefore carried out experiments to determine whether the model contributions are complementary.

We use 1 million sentences from CzEng 1.0 as our training data. We tune on WMT13 test set and we evaluate on WMT14 test set. Each BLEU score is an average over 5 runs of MERT. We use the feature set B in these experiments.

	LMs	
	stc	stc+tag
baseline	13.0	14.0
+source	**13.8**	**14.5**
+target	**14.0**	**14.7**

Table 7.5: Effect of LMs over morphological tags. Statistically significant differences are shown in bold.

As Table 7.5 shows, the LM over morphological tags is very beneficial, adding a full BLEU point on top of the baseline. We gain another 0.5 BLEU by including DPTM with source-context features only. When we also use target-context features, the BLEU score goes up another 0.2 points. When we compare these last two systems using MultEval, we find that the difference is statistically significant (α=0.01).

The comparison with a system without the morphological LM shows that the effects are largely complementary. In this baseline setting, source-context features improve BLEU more (by 0.8 as opposed to 0.5) but the difference between source-only and source+target model remains similar, around 0.2 BLEU points. Again, this difference is statistically significant according to MultEval.

While the benefit of DPTM is somewhat reduced when we add the morphological LM, the difference in BLEU compared to the source-context only remains significant. We attribute this improvement to our expressive feature set.

We do not only model the sequence of morphological tags. For instance, each tag in the target-side context is combined with every word and tag in the proposed translation. When the proposed phrase consists of multiple words, these features can allow the model to directly connect a target-context tag with a tag at the end of the proposed phrasal translation. In contrast, the LM over morphological tags needs to estimate the probability of the full n-gram (without skipping any words in between), and consequently struggles with data sparsity.

Furthermore, DPTM uses bilingual LM features. These features (extracted jointly from tags and lemmas) capture the relationship between source-context and target-context words and morphology. Bilingual LMs were shown to significantly improve translation quality (Niehues et al., 2011) and this information is simply not available to the decoder without DPTM.

Manual Analysis

In this section, we focus on manually identifying how DPTM affects translation outputs. Recall from our discussion in Section 3.3 that we expect the source-context model to improve mainly lexical choice (lemma selection) and the target-context information to help with morpho-syntactic coherence of the outputs (surface form selection).

Figure 7.1 shows a sentence from the test set translated by all system variants. The baseline translation has two main problems: the verb "took place" is translated as "došlo" which is correct but the Czech verb has an unusual va-

input:	the most intensive mining took place there from 1953 to 1962 .
baseline:	nejvíce intenzivní těžba **došlo** tam z roku 1953 **, aby** 1962 .
	the_most intensive mining$_{nom}$ ***there_occurred*** *there from 1953*
	, in_order_to *1962 .*
+source:	nejvíce intenzivní **těžby místo** tam z roku 1953 do roku 1962 .
	the_most intensive ***mining***$_{gen}$ ***place*** *there from year 1953 until year 1962 .*
+target:	nejvíce intenzivní těžba probíhala od roku 1953 do roku 1962 .
	the_most intensive mining$_{nom}$ *occurred from year 1953 until year 1962 .*

Figure 7.1: An example sentence from the test set. Each translation has a corresponding gloss in italics. Errors are marked in bold (Tamchyna et al., 2016a).

lency frame: it is subjectless and the event which occurred has to be expressed by a prepositional phrase with the preposition "k". The phrase-based system is not capable of such syntactic transformations and simply translates the subject into nominative case (which *would* be correct for most other verbs).

The second issue is the baseline translation of "to". This particle is ambiguous in English and its translation varies depending on its current sense. In this case, it expresses a time span but the PBMT system translates it in the sense of purpose, roughly as "in order to".

The output of the system with source-context model included still mistranslates the main verb. However, the translation of "to" is fixed as the correct sense is selected. We attribute this change to the context information: the surrounding numbers are most likely years and the preposition "from" probably helps disambiguate the sense as well. However, there is still a subtle error present: the translation of "from" should not be "z" in this sense.

Finally, the addition of target-context features on top of the source-context model fixes all the problems in this case. The model pushes the PBMT system to choose a verb translation "probíhala" for which morpho-syntactic coherence can be maintained more easily (the translation of "mining" can simply remain as the subject of the sentence). The translation of "from" is also corrected to "od roku" which is more suitable in this case.

Figure 7.2 shows several other examples of how the addition of source- and target-context model on top of the baseline gradually corrects errors in the translation. In all of these cases, the final translation is fully accurate and grammatical.

Naturally, in many sentences, the effect of our models is far less prominent and sometimes, DPTM even prefers a worse translation than the one produced

input:	destruction of the equipment means that Syria can no longer produce new chemical weapons .
+source:	**zničením** zařízení znamená , že Sýrie již nemůže vytvářet nové chemické zbraně . ***destruction_of***$_{instr}$ *equipment means , that Syria already cannot produce new chemical weapons .*
+target:	zničení zařízení znamená , že Sýrie již nemůže vytvářet nové chemické zbraně . *destruction_of*$_{nom}$ *equipment means , that Syria already cannot produce new chemical weapons .*
input:	nothing like that existed , and despite that we knew far more about each other .
+source:	nic takového neexistovalo , a přesto jsme věděli daleko víc o **jeden na druhého** . *nothing like_that existed , and despite_that we knew far more about* ***one***$_{nom}$ ***on other*** .
+target:	nic takového neexistovalo , a přesto jsme věděli daleko víc o sobě navzájem . *nothing like_that existed , and despite_that we knew far more about each other .*
input:	the authors have been inspired by their neighbours .
+source:	autoři byli inspirováni **svých sousedů** . *the authors have been inspired* ***their***$_{gen}$ ***neighbours***$_{gen}$.
+target:	autoři byli inspirováni svými sousedy . *the authors have been inspired their*$_{instr}$ *neighbours*$_{instr}$.

Figure 7.2: Examples of sentences from the test set showing improvements in morphological coherence. Each translation has a corresponding gloss in italics. Errors are marked in bold (Tamchyna et al., 2016a).

by the baseline system. However, on average, DPTM seems to have a positive effect on translation quality, as evidenced by the consistent improvements in BLEU.

We cannot fully confirm our expectation that source-context information helps mainly with lexical choice whereas target-context features improve morphology and syntax of the translation. Overall, our impression is that even the source-context model mostly improves morpho-syntactic coherence and corrections of semantics are more rare. Target-context information further helps maintain overall agreement and coherence of the translations.

So far, we have only relied on automatic metrics to judge the differences in translation quality. We carried out two annotation experiments to evaluate whether these differences are also visible for humans. In each of these experiments, the annotator was presented with the source sentence and two translations: the baseline and an improved system. The evaluation was blind (translations in each instance were ordered randomly) and the annotator's task was simply to compare the overall translation quality, without any specific criteria such as lexical choice or morphological coherence.

We selected 104 random sentences from the test set translated by the full-sized systems listed in Table 7.3. BLEU scores of this sample were 15.08 (base-

line), 16.22 (source context), and 16.53 (source+target context). This confirms that our sample does not deviate from the full test set – while the absolute BLEU scores are all slightly lower, the differences between them correspond to the differences measured on the full corpus. This sample was identical for both experiments.

Setting	Equal	Baseline is Better	New is Better
+source	52	26	26
+target	52	18	**34**

Table 7.6: Results of manual evaluation.

In the first experiment, the annotator compared the baseline and the system with source-context model. Interestingly, even though the difference in BLEU is convincingly large (over 1 BLEU point in the sample), the annotator found no difference between the two systems. Table 7.6 shows the specific results.

In the second experiment, the annotator compared the baseline and the system with source- and target-context model. In this setting, the improved system was judged as better in roughly 1/3 of the instances, both were marked as equal quality in half of the instances and in the remaining 18 sentences, the baseline translation won.

Integration in Chimera

The improvement of the state of the art for English-Czech MT is our ultimate goal. We therefore integrate DPTM into Chimera for WMT16. Our baseline is thoroughly described in Section 3.1. Let us only note here that the main phrase table is trained on the new CzEng 1.6 corpus. This new version is several times larger than CzEng 1.0 which we use for our other experiments.

Our model setup is somewhat cumbersome in this setting. First, we do *not* use CzEng 1.6 as the training data for the model. In preliminary experiments for WMT16, it was found that using CzEng 1.6 may not lead to better BLEU as opposed to using CzEng 1.0. Coupled with the difference in size, we opt to use CzEng 1.0 for DPTM. Second, because training the full Chimera system is very time consuming, we attempt to reuse as much of the baseline system as possible. We therefore do not build an additional phrase table with the factors required by DPTM and instead reduce the feature set only to stc, lemma and tag factors on each side. Our feature set is a subset of Czech A.

Chimera is a highly tuned MT system so it is a very difficult baseline to beat. In fact, we have experimented with various techniques to improve Chimera

in the past and many of them failed, even though they usually provide good results. For instance, hierarchical models or lexicalized reordering models do not improve BLEU for Chimera.

We show the results in Table 7.7. The first four rows show BLEU scores of the basic PBMT component of Chimera where we always add a single model. Each BLEU score is again an average over 5 optimization runs. DPTM improves over the PBMT baseline significantly (from 19.1 to 19.4) but not as much as OSM or TectoMT.

setup	**BLEU**
M	19.1
M+DPTM	19.4
M+OSM	19.7
M+T	20.0
M+T+OSM	20.91±0.67*
M+T+OSM+DPTM	20.96±0.67*

Table 7.7: BLEU scores of Chimera system variants in 2016. Results obtained from a single optimization run are marked by asterisks.

The last two rows are the results of a single optimization run as reported in Tamchyna et al. (2016b). They show that unfortunately, DPTM does not improve translation quality of the final system. This is further supported by the results of the manual evaluation which ranked the two systems (Chimera, Chimera+DPTM) as identical.

TectoMT is a very strong component which provides the PBMT baseline with long, (mostly) grammatical phrases. Long-distance dependencies which are important for morphological coherence are handled well in TectoMT (which works with deep syntactic parse trees instead of sequences of surface forms). DPTM arguably does not have as much space for improving morpho-syntactic coherence when the TectoMT phrase table is included in the system.

Errors in Morphology

Recall our automatic analysis of morphological errors made by the various components of Chimera in Section 3.2. We made several observations about the types of errors that TectoMT can improve and errors that all of the components struggle with. We repeat the analysis here for the system with DPTM and look for a systematic improvement in some of these categories.

Table 7.8 does not provide a clear answer. DPTM seems to slightly reduce the number of errors in morphology across most parts of speech. The dif-

System	# lemmas	# errors	# errors by part of speech				
			A	C	N	P	V
M	**29163**	**4600**	819	98	2019	434	1230
M+DPTM	29101	4458	823	105	1959	416	1155

Table 7.8: Morphological errors made by Chimera and DPTM divided by part of speech. A=adjective, C=numeral, N=noun, P=pronoun, V=verb.

ferences seem too small to suggest any significant pattern. Unfortunately, we do not find any systematic improvement of a certain error type.

7.3 Model Training

In this section, we describe experiments related to how we set the parameters of our model and to model training.

Intrinsic Evaluation

We have already presented some results of intrinsic evaluation without clearly defining how we measure it. Here we evaluate DPTM in three basic settings and present the results.

We evaluate DPTM by measuring accuracy, i.e. the ratio of correctly classified instances.[4] Recall that our basic setting is multi-class classification: each source phrase has a number of possible translations and the task of DPTM is to predict which one is correct in the current (source and target) context.

In order to estimate the model accuracy, we use a development set. We extract all possible phrase pairs from this set and we generate instances for classification from each phrase pair occurrence and its context (exactly as we would do in training). We then use a trained model to predict the conditional probability distribution over the possible translations of each source phrase in the held-out set. The model prediction is correct when the correct translation is the most probable one.

We also define a baseline to compare against. For each source phrase, we look at the phrase-level translation probability $P(\bar{e}|\bar{f})$. The baseline predicts the most probable translation every time regardless of the current context (this is equivalent to always choosing the most frequent translation).

[4]For readability, we present accuracy multiplied by 100 throughout this work.

For this experiment, we use English-Czech translation again. We train all models on the full CzEng 1.0 corpus and we evaluate them on WMT13 test set. The baseline achieves an accuracy of 51.5. This number is quite high considering the morphological richness of Czech. The source-context only model reaches 66.3 and when we also add target-context features, the accuracy improves to 74.8.

The final number is not really comparable with the previous two results because of how we generate the instances: the last model needs target-side context information but because this evaluation happens outside of MT decoding, we have no partial MT outputs. We therefore extract the target-context features from the *true* target sentence, which gives the target-context model an unfair advantage. However, the result does show that there is potential in the target context features.

Regularization

We experimented with various settings for L^2 regularization. The motivation for using a regularization term is to avoid model overfitting by penalizing high values in the weight vector. The aim is to prevent the model from relying on any single feature too heavily.

L^2 regularization term is simply the L^2 norm of the parameter vector $\boldsymbol{w}$. The term is added to the loss function with a weight λ_{L^2}; the weight can be tuned to optimize the model accuracy on a held-out set.

In terms of implementation in VW, note that the regularization weight is interpreted differently depending on whether VW is in the online learning mode or in a batch scenario. In our case (online learning), the term is applied for each example; the L^2 weight therefore needs to be set rather low.

As illustrated in Figure 7.3, we found no setting of the weight which improves either the classifier accuracy or the final BLEU score. As the weight approaches zero, the effect of the regularization is so low that it is effectively not applied.

The smallest value that we evaluate is 1×10^{-9} where the obtained accuracy is 78.47 and BLEU is 13.62$\pm$0.45, i.e. nearly identical to our baseline where L^2 regularization is disabled.

Loss of Negative Examples

All of our experiments in this work are carried out with the 0/1 loss scheme in training because in our preliminary evaluation, the BLEU loss did not perform

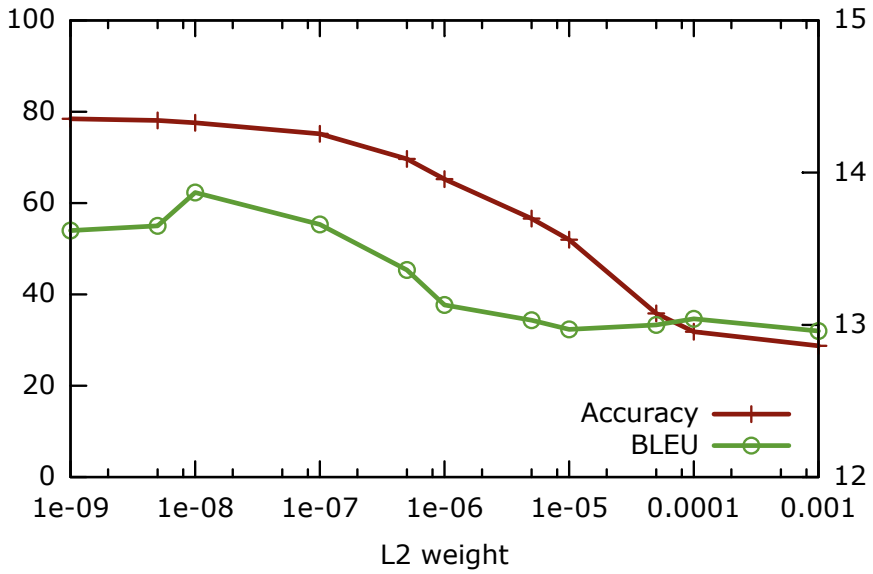


Figure 7.3: Effect of L^2 regularization on model accuracy and BLEU. The baseline accuracy is 78.55 and BLEU is 13.81.

better. In this section, we compare the two settings more rigorously in a full MT system pipeline.

We use our medium-size data setting with 5 million parallel training sentences. For both loss types, the average BLEU of the WMT14 test set over several runs of MERT is 16.2. We find no statistically significant difference between the two ways of penalizing negative examples. This finding is somewhat surprising: we have expected that changing model training is such a prominent way would have a significant effect on final results.

There is still room for experimentation in this area. Notably, sentence-level BLEU has repeatedly shown a low correlation with human ranking (see e.g. Bojar et al., 2016c). The metric also disregards word similarity (inflections of the same lemma are penalized the same as an entirely different word). Experiments with more recent metrics might lead to better results.

Leaving One Out and Phrase Table Features

As described in Section 5.2, our original aim was to include translation model scores as features in DPTM. In preliminary experiments, we did not find these features to be helpful and we therefore use a simpler feature set in the majority of our experiments. In this section, we examine the effect of using TM scores

in the model both in terms of intrinsic and extrinsic evaluation. We also look at the interplay between our leaving-one-out strategy and TM features.

Leave	**Feature Set**	
One Out	standard	+TM scores
×	75.40	74.71
✓	**76.05**	75.12

Table 7.9: Model accuracy with TM features and leaving one out.

We show in Table 7.9 that in terms of classifier accuracy, leaving one out is always helpful. Somewhat surprisingly, the difference is larger when TM features are *not* included. As discussed in Section 5.3, leaving one out affects feature extraction even when TM features are not used because it filters out some of the training instances.

First recall that we also use phrase table filtering. Our setting of this filter always discards singleton phrase pairs. These are "1-1-1" singletons: $c(\bar{e}), c(\bar{f})$ and $c(\bar{e} \wedge \bar{f})$ are all equal to one. (I.e. neither the source nor the target phrase appear anywhere else in the data.) In order for the phrase pair to appear in the filtered phrase table, at least one of $c(\bar{e}), c(\bar{f})$ therefore must be higher than one.

Leaving one out will then discard a phrase pair if and only if $c(\bar{e} \wedge \bar{f}) = 1$. These are phrases that appear multiple times in the training data but co-occur only once. In our experiments, about 4.4 per cent of training examples are discarded by this criterion. Because their filtering improves held-out accuracy, we believe that these phrase pairs are mostly noisy and/or do not generalize well outside their specific context.

Another result illustrated by Table 7.9 is that we achieve better accuracy when not using the TM scores. Unfortunately, it seems that even with significance test filtering and leaving one out, we cannot entirely avoid overfitting to these features.

Several factors may contribute to this result. First, because TM scores (unlike all other features in our feature set) are *dense* features – they appear with every training example – their weight grows naturally larger during the online training of our model and could be overestimated. We do use the default *adaptive* setting of VW which maintains different learning rates for individual features and theoretically should allow the mixing of dense and sparse features. However, a more careful treatment of these dense score features might be necessary.

Second, the phrase table still captures the probability distribution of the training data; when the model learns to rely on it too heavily, it may not generalize to our dev/test domain as well as before.

Table 7.10 shows the final BLEU scores of the evaluated configurations. We find that even though leaving one out has some effect on the model accuracy, the difference is not apparent in translation quality when TM features are not used.

When we add the TM score features, the overall BLEU score decreases significantly. As expected, leaving one out seems to play a role in this setting. It gives a slight improvement (15.8 vs. 15.9) but this difference is not statistically significant according to MultEval (p-value is 0.03).[5]

Leave	**Feature Set**	
One Out	standard	+TM scores
×	16.2	15.8
✓	16.2	15.9

Table 7.10: Final BLEU scores with TM features and leaving one out.

An interesting observation is that model accuracy and BLEU score are not necessarily correlated. When TM scores are disabled, we obtain a 0.65 improvement in accuracy thanks to leaving one out but there is no effect on BLEU. Similarly, the difference between accuracies 75.40 and 75.12 is small but there is a 0.3 difference in BLEU between these systems. Insight into the feature set of the model seems essential for understanding its performance in the MT system.

7.4 Additional Language Pairs

So far, we have focused on English-Czech translation. However, at its core, our method is language independent. While the feature set that we have presented is tailored more to translating into morphologically rich languages, DPTM can work with any target language. Similarly, while we require some linguistic tools for both the source and the target language, DPTM can conceivably improve translation even when such tools are not available (see our discussion in Section 5.2).

[5]Note that we report BLEU=16.4 in our main English-Czech translation results. This difference illustrates the degree of randomness in MERT (even when we do multiple runs and averaging), here a different sample lead to a slightly worse result.

Feature Type	Configurations	
	German	Romanian
Source Indicator	s, l, l+t, t	l, t
Source Internal	s, s+a, s+p, l, l+t, t, a+p	l, l+a, l+p, t, a+p
Source Context	s (-3,3), l (-3,3), t (-5,5)	l (-3,3), t (-5,5)
Target Context	s (2), l (2), t (2), l+t (2)	l (2), t (2)
Bilingual Context	l+t/l+t (2)	l+t/l+t (2)
Target Indicator	s, l, t	l, t
Target Internal	s, l, l+t, t	l, t

Table 7.11: Feature templates used for English-German and English-Romanian translation. Letter abbreviations refer to word factors: s (stc), l (lemma), t (morphological tag), a (analytical function), p (lemma of dependency parent). Numbers in parentheses indicate context size.

In this section, we present our experiments with several other language pairs. In the next section, we evaluate DPTM for English-German and English-Romanian translation. We then move to translation into English with Czech, German and Romanian as source languages.

English-German

We have described the linguistic tools used for processing these languages in Section 5.2. In terms of corpora, we train on the constrained parallel data from the WMT14 translation task. The data consists of Europarl (Koehn, 2005) and Common Crawl[6] corpora and amounts to around 4.3 million sentences. We tune and evaluate our systems on WMT13 and WMT14 test sets, respectively.

The feature set for English-German translation is different from English-Czech, Table 7.11 lists the used feature templates.

Otherwise, our setup is fairly similar to English-Czech experiments: we have a 5-gram LM over surface forms trained on the target side of the parallel data. We tune the system using MERT and report the average over 5 runs.

The baseline BLEU score average is 15.7 and the addition of DPTM (source and target context) improves this result to 16.2; the difference is statistically significant. While this improvement is not as large as we have observed for English-Czech translation, it still shows that DPTM can be used for a very different target language in a straightforward way.

[6] http://commoncrawl.org/

Note that because the German tagset is much more coarse-grained than the Czech tagset, the improvements in morphological coherence may be lower than would be possible with a more suitable granularity of tags.

English-Romanian

We report on two different sets of experiments with English-Romanian. The first experiment has a very similar setting to our evaluation of English-Czech and English-German translation. The second set was carried out as part of the WMT16 translation task. Both experiments use the same feature set which is described in Table 7.11.

The parallel training data are identical in both experiments: we use the data made available for WMT16, i.e. Europarl and SETIMES2 (Tiedemann, 2009), around 600 thousand sentence pair altogether.

In the first experiment, the baseline system is fairly similar to our other baselines: the LM is a 5-gram model trained only on the target side of the parallel data. We tune and test our system on WMT16 dev- and test-sets, respectively.

The baseline system achieves a BLEU score of 19.6. When we add DPTM (again, both with source and target context information), the BLEU score improves to 20.2 points. This difference is statistically significant.

Our second set of experiments is related to a system-building effort for the WMT16 translation task. Here we do not attempt to carefully isolate and describe the effect of DPTM. Instead, we are interested simply in developing the best possible system for the language pair at hand. These experiments are described in more detail in Tamchyna et al. (2016b).

Setting	**BLEU**
baseline	26.2
+tagLM	**26.6**
+ccrawl	**28.0**
+RM	28.1
+DPTM	**28.3**

Table 7.12: BLEU scores of system variants for English-Romanian translation (Tamchyna et al., 2016b).

Because these experiments were carried out before WMT16 ended (the reference translation of the test set was not available), we split the provided development set into two parts; we tune on the first half and evaluate on the second

half (so the scores are not comparable between the first and the second set of experiments).

Our baseline uses a LM trained not only on parallel data but also on the News Crawl 2015 data provided in WMT. Table 7.12 lists the system results (with statistically significant differences marked in bold) as we gradually include additional components in the system: a 7-gram LM over morphological tags (+tagLM), a large 4-gram LM over surface forms trained on the Common Crawl corpus (+ccrawl), a lexicalized reordering model (+RM) and finally, our discriminative model with source and target-side context information (+DPTM).

The results show that DPTM can improve translation into other languages and that it can outperform even a very competitive baseline. Note that some of the models included in the baseline are trained on much larger data than DPTM (which does not use the large additional monolingual data).

Translation into English

In this section, we describe our experiments with translation into English. We developed DPTM and the features mainly with morphologically rich target languages in mind. Morpho-syntactic coherence in English is more driven by word position than inflection, DPTM therefore does not have as much space for improvement in this scenario. Also recall our manual analysis in Section 7.2 – in English-Czech translation, DPTM seems to improve morpho-syntactic coherence more than semantics overall. If we reduce its ability to improve morphology, we can expect the gains to be lower.

We evaluate three language pairs: German-English, Romanian-English and Czech-English. Let us first describe the feature set for each language pair and then move on to the data sets that we use.

Table 7.13 lists the features used for each setting. We do not use a dependency parser for German and Romanian. This limits our feature set to forms, lemmas and tags on each side. For Czech-English, we use the MST parser trained on the Prague Dependency Treebank 2.0 (Hajič et al., 2006). Analytical functions (e.g. "Sb" for subjects) could arguably provide some information to the model about the required *position* of English words in the translation.

The data used for German-English are identical to our English-German experiments. For Romanian-English, the data are the same as for the WMT system-building effort. Our Czech-English data are identical to the opposite direction, we use the full CzEng 1.0 corpus. The evaluation sets therefore differ

Feature Type	Configurations	
	German, Romanian	Czech
Source Indicator	l, t	l, t
Source Internal	l, t	l, l+a, l+p, t, a+p
Source Context	l (-3,3), t (-5,5)	l (-3,3), t (-5,5)
Target Context	l (2), t (2)	l (2), t (2)
Bilingual Context	l+t/l+t (2)	l+t/l+t (2)
Target Indicator	l, t	l, t
Target Internal	l, t	l, t

Table 7.13: Feature templates used for German-English, Romanian-English and Czech-English translation. Letter abbreviations refer to word factors: s (stc), l (lemma), t (morphological tag), a (analytical function), p (lemma of dependency parent). Numbers in parentheses indicate context size.

among the languages: we use WMT14 for German and Czech and the second half of WMT16 development set for Romanian.

	De-En	**Ro-En**	**Cs-En**
baseline	22.2	28.4	22.7
+DPTM	**22.4**	**28.7**	**23.3**

Table 7.14: Results of experiments with translation into English.

Table 7.14 lists the results. As expected, the improvements are smaller overall than in the opposite direction. However, DPTM does consistently and significantly improve translation quality even in this setting. The difference is the largest in Czech-English, possibly thanks to the source-side syntactic features.

7.5 Source Context for Target-Side Morphology

In this section, we describe a small annotation experiment which we carried out to determine whether target-side morphology can be successfully predicted from source context information only. This was a preliminary experiment and we present its results here for completeness.

We already illustrated the need for target-side context when predicting word inflections on several examples in Section 3.3. Here we ask human annotators to guess the correct morphological variant of target-side words and we quantify the results. Our motivation for this experiment is to set an upper bound on the prediction accuracy of target-side morphology when only

source-context information is available – if even a human annotator cannot disambiguate all instances correctly, we cannot realistically expect a machine-learning model to successfully solve this task.

Specifically, the annotation task was to determine the surface form of a Czech word given the following information:

- the word lemma and POS,
- the full source sentence (English) along with its morphological analysis,
- and the set of English words aligned to the predicted Czech word.

Factor	**Source Sentence**							
form	He	wants	me	to	call	my	**mother**	.
lemma	he	want	me	to	call	my	mother	.
tag	PRP	VBZ	PRP	TO	VB	PRP$	NN	.
afun	Sb	Pred	Sb	AuxV	Adv	Atr	Obj	AuxK

Target word: matka|N

Figure 7.4: Example instance of the annotation task for morphology prediction. The annotator is asked to write the correct form of the word "matka" given its POS and the source sentence. The correct inflection in this case is "matce".

Figure 7.4 shows an example instance for annotation. In this instance, the annotator should predict the surface form of the lemma "matka" ("mother") in the translation. The correct answer is extracted from the target side of the parallel data; the annotator does not see the Czech translation and has to *imagine* what the translation probably looks like. In this instance, the actual translation contains the verb "zavolat" which requires a dative case for the object: "matce".

This setting may seem unnatural but the information that the annotators see is roughly what the classifier has access to in the features. Naturally, annotators use their intuition – both linguistic and based on their world knowledge – while the classifier only has a model trained on some limited corpus.

Two independent annotators were presented with 100 instances. The annotation instances were randomly selected from parallel sentences taken from CzEng 1.0. We discarded 13 instances which were marked as erroneous by one or both annotators. Causes of these errors were two-fold: either the predicted word did not belong in the sentence at all (implying a non-parallel sentence pair or a very loose translation) or it was misaligned within the English sentence.

From the remaining 87 instances, annotator A answered correctly in 67 cases and annotator B in 65 cases. They agreed in 65 instances (not only the correct ones). While a more rigorous study would be needed to obtain a reliable estimate, our small annotation experiment suggests that human annotators have roughly 0.76 accuracy on the filtered instances and 0.66 on all instances. The inter-annotator agreement is around 0.74.

Note that this experiment was done on real parallel data (albeit automatically collected and therefore noisy). Results using MT outputs could be somewhat different.

We can interpret this result in two ways. It could be viewed as a positive result for using only source context – if 76% accuracy was attainable with a machine learning model, this would most likely be a useful signal for the MT system already. Indeed, source context information has been used for predicting morphology quite successfully, as discussed in Chapter 8.

On the other hand, the result shows that there is an inherent limit on what is achievable with source context information alone. This, along with our observations in Section 3.3 confirms the strong motivation for considering also target-side context information in morphology prediction.

8

Related Work

There is a large amount of work which focuses on utilizing wider source-context information in MT, mostly by applying discriminative models. Here we describe several of the most relevant articles from this area.

Giménez and Màrquez (2007) proposed to use source context for phrasal translation. They incorporated their module in a full MT system, thus being able to evaluate its impact on translation quality. While no improvement of BLEU score was achieved, they carried out manual evaluation and confirmed that their model helps MT quality. They used a rich feature set based on the WSD system of Yarowsky et al. (2001). Local context was captured by a window of fixed length 5, from which words, POS tags and phrase-chunking labels were extracted. Global context was modeled as a bag of lemmas of the whole source sentence.

They used local linear support vector machines (SVM). For each source phrase $\bar{f}_i$ and for each of its possible translations $\bar{e}_j$, they trained a binary one-against-all classifier $C_{i,j}$. Sentences where $\bar{f}_i$ was translated as $\bar{e}_j$ served as positive examples and all other occurrences of $\bar{f}_i$ as negative examples. Training such models is very taxing in terms of computing time and storage space and does not allow for parameter sharing. No target context information was used.

Carpuat and Wu (2007) described an approach related to Giménez and Màrquez (2007), naming the task "phrase sense disambiguation" (PSD). The PSD module was based on their WSD system (Carpuat et al., 2004) and utilized an ensemble of four machine-learning models: a naive Bayes model, a maximum-entropy model, a boosting model and a kernel-PCA model. The features were also based on their state-of-the-art WSD system. They carried

out a large-scale evaluation using realistic data sizes across several language pairs. Their results showed a consistent improvement of scores according to a wide range of MT metrics (including BLEU). This approach has a similar drawback to Giménez and Màrquez (2007): predictors for different source phrases are trained independently. The method was never fully integrated in phrase-based search and it does use target-side context.

Chan et al. (2007) included source-context information in a hierarchical MT system. They used SVMs to predict the probability of translation given source context and included the scores as a feature in the decoder (limiting the predictions to short phrases). They also reported gains in BLEU score.

Mauser et al. (2009) introduced the "discriminative word lexicon", a technique which attempts to predict, for each word in the target vocabulary, whether or not it should appear in the translation of a given sentence. The authors trained a binary classifier for each target word, using as features only the bag of words (from the whole source sentence). Sentences where the target word occurred were used as positive examples, other sentences served as negative examples. During decoding, all classifiers were queried and translation hypotheses were rewarded based on the scores of words that they contained. Significant improvements of BLEU score were reported.

Jeong et al. (2010) also proposed a discriminative lexicon with features extracted from the context. Their feature set was designed for morphologically rich languages but they only make use of source context information.

Daiber and Sima'an (2015) make an explicit assumption that target-side morphology can be partially predicted by the source-side context only. They show that indeed, source-context information can be used successfully to some degree.

Subotin (2011) proposed a discriminative model and unlike others, they also included target-side context information. However, the algorithm is tightly coupled with cube pruning in hierarchical MT and includes specific rules for passing non-local information about agreement between the individual rules. Our method provides a more general framework where any information can be extracted from source- and target-side context. While we do rely on hypothesis construction from left to right, we do not make assumptions about the pruning strategies used.

Discriminative LMs are also relevant to our work: they use non-local features from (only) the target-side context in a discriminative setting. Notably, inspired by the success of a similar technique in automatic speech recognition (ASR), Li and Khudanpur (2009) proposed a discriminative LM for rescoring MT outputs. Unlike our method, these LMs are not integrated in search but are

only used for n-best re-ranking. However, this enables them to use arbitrary target-side features because the full translation is already known.

Our idea for using bilingual context features comes from the work on bilingual LMs (Niehues et al., 2011) where the authors extended standard n-gram LMs to also capture limited source-side context. This is achieved by concatenating each target word in the LM context window with its source-side counterpart(s). While this technique indeed allows the system to condition phrasal translations on the source-side context, the model may suffer from data sparsity inherently present in count-based n-gram LMs.

Finally, neural networks have become a prominent topic in MT recently. Our work is most similar to Devlin et al. (2014) who add the predictions of a fully-connected multilayer NN as an additional feature in a phrase-based MT system. The NN has access to both source-side and target-side context. Our approach is arguably more flexible in the design of features and may allow more expressive feature sets to be easily integrated in the model – application-specific features (e.g. promoting correct translations of fixed terminology) may easily be added to our model; similarly, the addition of discourse-level information or sentiment annotation is trivial in our model. Devlin et al. (2014) describe a number of technical challenges involved in optimizing the model speed for realistic use; our obstacles were somewhat different and our solutions may be useful to other researchers.

Purely neural MT based on the encoder-decoder architecture (Sutskever et al., 2014) is the newly emerging state of the art in MT. This approach can theoretically use information from arbitrarily long context (source- and target-side); indeed, translations produced by IWSLT2015 NMT submissions show that NMT systems usually outperform PBMT in grammaticality and long-distance linguistic dependencies between words (Bentivogli et al., 2016). On the other hand, PBMT still has a strong presence in the industry and thanks to the consistent improvements of translation quality provided by DPTM and also the number of possible applications that DPTM allows, our model may still be useful to both researchers and users of MT in the future.

9 Conclusion

The thesis describes our attempt to address the issues of lexical and morphological choice in statistical MT. We developed a phrasal discriminative translation model (DPTM) with a rich set of features extracted from both source- and target-side context. The model is fully integrated in phrase-based search and can be used relatively efficiently even on large data sets.

DPTM leads to consistent improvements of translation quality over baseline PBMT systems in various settings. Importantly, the improvements from both the source- and target-context model scale well as the training data grows larger, suggesting that DPTM indeed provides the MT system with information which is not available to the baseline model components. Also, even though we primarily designed DPTM and its features for English-Czech translation, the whole framework is language-independent and can be successfully applied to other language pairs. We have provided several analyses of the results, both automatic and manual, to illustrate what types of phenomena are improved by our model.

However, when integrated in our strongest baseline, Chimera, DPTM does not provide any gains in BLEU or according to human evaluation. We believe that DPTM may potentially still be useful for Chimera in different settings: for instance, our model could be used to capture extra-sentential context information which Chimera currently does not utilize. We have not experimented with such settings in this work.

We have not described all of our attempts to improve translation quality using discriminative modelling. In particular, we invested a lot of effort in training on n-best lists with the following motivation: some authors have claimed that discriminative models only work well when trained towards the final eval-

uation metric (such as BLEU), not cross-entropy loss (see Auli et al. 2014, inter alia). Also, by training on n-best lists, the model should theoretically learn to fix errors that the baseline system actually makes and that are correctable (Li and Khudanpur, 2009). Moreover, exposure bias might be mitigated by training on n-best lists (in our training scheme, target-side features of our model are extracted from the *true* target context, perhaps leading the model to rely on them too much). However, our work in this direction was not successful and when our simple training scheme (train on true parallel data, optimize cross-entropy) began to produce positive results, we did not explore this area further.

We also helped implement and experimented with a word-level discriminative model, originally proposed as a team project for the MT Marathon in Prague in 2013: "A Discriminative Lexicon for Translating to Morphologically Rich Languages".[1] We even integrated the model in Chimera for a WMT submission but due to the lack of positive results, we eventually focused entirely on DPTM. It may in fact be interesting to reconsider the word-level model as an alternative now that the implementation of DPTM is finished and thoroughly evaluated.

9.1 Future Work

DPTM provides quite a general framework for using discriminative models in PBMT. With small extensions of the feature set, DPTM could be used to take into account various new types of information.

For instance, by including features describing the surrounding sentences, DPTM could access discourse-level information and perhaps be used to improve MT output coherence when translating longer documents.

Similarly, by explicitly adding features which describe the sentiment of the input sentence, DPTM might be used to steer the MT output towards translations with a similar emotional polarity.

DPTM is already being used successfully within the European innovation project Health in my Language (HimL) to explicitly model verb valency and several other phenomena related to semantic fidelity.[2]

In the following section, we describe one particular application which is beyond the scope of the thesis but is already completed and published.

[1] `http://ufal.mff.cuni.cz/mtm13/project-final-presentations/lexicon_morph_rich_final.pdf`

[2] `http://www.himl.eu/files/D2.2-intermediate-report-on-srl-and-fidelity.pdf`

Producing Unseen Morphological Variants in SMT

This work was done jointly with LMU Munich and is described in Huck et al. (2017). The idea and motivation for this application of DPTM was proposed by Alexander Fraser, who also proposed the design of the feature set for this setting.

MT systems struggle when translating into morphologically rich languages with a large target-side vocabulary. Particularly, while the system may know the correct lemma of the word to be translated, its inflected surface form required in the given situation may be unseen in the training data. The work that we describe here aims to enable the MT system to produce any inflected word form of known translations.

We address this issue by using a morphological generator for the target language and by adding unknown inflections of words as new (synthetic) entries in the MT system phrase table. We focus on phrases with a single word on the target side (to avoid problems with agreement). For each phrase, we obtain the lemma of the target-side word and then generate all its inflected word forms (optionally with some constraints). We then add all these new word forms as additional translations of this phrase. We focus on Czech and use MorphoDiTa (Straková et al., 2014) to perform the generation.

The main issue that the work addresses is how to score the synthetic phrase pairs. Phrase translation probabilities and lexical weights are based on counts which are all zero for unseen word forms.

Our contribution to this work is the use of DPTM. We divide the set of features into two disjoint parts. The first part only describes the *lexical* translation (disregarding morphology) and the second part only describes the morphological properties of words (and disregards the lexemes). By splitting our feature set this way, we effectively obtain two separate models: a predictor of lexical choice and an independent predictor of target-side morphology. When DPTM scores a previously unseen word form, there is effectively no difference from scoring a known surface form: the lemma and tag are viewed independently and both are known to the model. There is similarity between this work and two-step translation (Bojar and Kos, 2010; Fraser et al., 2012) which handles morphological inflection as a second step after translation into stemmed words – we can view our approach as "two-step MT in one step".

This work improves the quality of MT into Czech in small- and medium-resource settings and significantly reduces the target-side OOV rate. It represents a successful application of our method in a novel setting.

Bibliography

Agarwal, A. – Chapelle, O. – Dudík, M. – Langford, J. A Reliable Effective Terascale Linear Learning System. *CoRR*. 2011, abs/1110.4198. Available at: http://arxiv.org/abs/1110.4198.

Auli, M. – Galley, M. – Gao, J. Large-scale Expected BLEU Training of Phrase-based Reordering Models. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, p. 1250–1260, 2014. Available at: http://aclweb.org/anthology/D/D14/D14-1132.pdf.

Bentivogli, L. – Bisazza, A. – Cettolo, M. – Federico, M. Neural versus Phrase-Based Machine Translation Quality: a Case Study. *CoRR*. 2016, abs/1608.04631. Available at: http://arxiv.org/abs/1608.04631.

Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006. ISBN 0387310738.

Bojar, O. – Tamchyna, A. CUNI in WMT15: Chimera Strikes Again. In *Proceedings of the 10th Workshop on Machine Translation*, p. 79–83, Stroudsburg, PA, USA, 2015. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-941643-32-7.

Bojar, O. – Tamchyna, A. The Design of Eman, an Experiment Manager. *The Prague Bulletin of Mathematical Linguistics*. 2013, 99, p. 39–58. ISSN 0032-6585.

Bojar, O. – Buck, C. – Federmann, C. – Haddow, B. – Koehn, P. – Leveling, J. – Monz, C. – Pecina, P. – Post, M. – Saint-Amand, H. – Soricut, R. – Specia, L. – Tamchyna, A. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, p. 12–58, Baltimore, MD, USA, 2014. Association for Computational Linguistics. ISBN 978-1-941643-17-4.

Bojar, O. – Chatterjee, R. – Federmann, C. – Haddow, B. – Huck, M. – Hokamp, C. – Koehn, P. – Logacheva, V. – Monz, C. – Negri, M. – Post, M. – Scarton,

C. – Specia, L. – Turchi, M. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the 10th Workshop on Machine Translation*, p. 1–46, Stroudsburg, PA, USA, 2015. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-941643-32-7.

Bojar, O. et al. Findings of the 2016 Conference on Machine Translation (WMT16). In Bojar, O. – . (Ed.) *Proceedings of the First Conference on Machine Translation (WMT). Volume 2: Shared Task Papers*, 2, p. 131–198, Stroudsburg, PA, USA, 2016a. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-10-4.

Bojar, O. – Dušek, O. – Kocmi, T. – Libovický, J. – Novák, M. – Popel, M. – Sudarikov, R. – Variš, D. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In Sojka, P. – Horák, A. – Kopeček, I. – Pala, K. (Ed.) *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, č. 9924 v Lecture Notes in Computer Science, p. 231–238, Cham / Heidelberg / New York / Dordrecht / London, 2016b. Masaryk University, Springer International Publishing. ISBN 978-3-319-45509-9.

Bojar, O. – Kos, K. 2010 Failures in English-Czech Phrase-based MT. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, p. 60–66, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. Available at: `http://dl.acm.org/citation.cfm?id=1868850.1868855`. ISBN 978-1-932432-71-8.

Bojar, O. – Buck, C. – Callison-Burch, C. – Federmann, C. – Haddow, B. – Koehn, P. – Monz, C. – Post, M. – Soricut, R. – Specia, L. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, p. 1–44, Sofia, Bulgaria, August 2013a. Association for Computational Linguistics. Available at: `http://www.aclweb.org/anthology/W13-2201`.

Bojar, O. – Graham, Y. – Kamran, A. – Stanojević, M. Results of the WMT16 Metrics Shared Task. In *Proceedings of the First Conference on Machine Translation*, p. 199–231, Berlin, Germany, August 2016c. Association for Computational Linguistics. Available at: `http://www.aclweb.org/anthology/W/W16/W16-2302`.

Bojar, O. – Žabokrtský, Z. – Dušek, O. – Galuščáková, P. – Majliš, M. – Mareček, D. – Maršík, J. – Novák, M. – Popel, M. – Tamchyna, A. The Joy of Parallelism with CzEng 1.0. In Chair), N. C. C. – Choukri, K. – Declerck,

T. – Dočan, M. U. – Maegaard, B. – Mariani, J. – Moreno, A. – Odijk, J. – Piperidis, S. (Ed.) *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Bojar, O. – Rosa, R. – Tamchyna, A. Chimera – Three Heads for English-to-Czech Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, p. 92–98, Sofia, Bulgaria, August 2013b. Bălgarska akademija na naukite, Association for Computational Linguistics. Available at: http://www.aclweb.org/anthology/W13-2208.

Braune, F. – Fraser, A. – Daumé III, H. – Tamchyna, A. A Framework for Discriminative Rule Selection in Hierarchical Moses. In Bojar, O. – . (Ed.) *Proceedings of the First Conference on Machine Translation (WMT). Volume 2: Shared Task Papers*, 2, p. 92–101, Stroudsburg, PA, USA, 2016. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-10-4.

Brown, P. F. – Pietra, S. A. D. – Pietra, V. J. D. – Mercer, R. L. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*. June 1993, 19, 2, p. 263–311.

Carpuat, M. – Wu, D. Improving Statistical Machine Translation using Word Sense Disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Prague, Czech Republic, 2007.

Carpuat, M. – Su, W. – Wu, D. Augmenting Ensemble Classification for Word Sense Disambiguation with a Kernel PCA Model. In *Proceedings of Senseval-3, Third International Workshop on Evaluating Word Sense Disambiguation Systems*, Barcelona, Spain, 2004. SIGLEX, Association for Computational Linguistics.

Carpuat, M. – Daumé III, H. – Fraser, A. – Quirk, C. – Braune, F. – Clifton, A. – Irvine, A. – Jagarlamudi, J. – Morgan, J. – Razmara, M. – Tamchyna, A. – Henry, K. – Rudinger, R. Domain Adaptation in Machine Translation: Final Report. Technical report, 2012. Available at: http://hal3.name/damt/.

Chan, Y. S. – Ng, H. T. – Chiang, D. Word Sense Disambiguation Improves Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, p. 33–40, Prague, Czech Republic, June 2007. Association for Computational Linguistics. Available at: http://www.aclweb.org/anthology/P/P07/P07-0005.

Chen, S. F. – Goodman, J. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*. 1999, 13, 4, p. 359–393. Available at: http://dx.doi.org/10.1006/csla.1999.0128.

Cherry, C. – Foster, G. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, p. 427–436, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. Available at: http://dl.acm.org/citation.cfm?id=2382029.2382089. ISBN 978-1-937284-20-6.

Clark, J. H. – Dyer, C. – Lavie, A. – Smith, N. A. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, p. 176–181, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. Available at: http://dl.acm.org/citation.cfm?id=2002736.2002774. ISBN 978-1-932432-88-6.

Daiber, J. – Sima'an, K. Machine Translation with Source-Predicted Target Morphology. In *Proceedings of the 15th Machine Translation Summit (MT Summit 2015)*, p. 283–296, 2015.

Devlin, J. – Zbib, R. – Huang, Z. – Lamar, T. – Schwartz, R. M. – Makhoul, J. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, p. 1370–1380, 2014. Available at: http://aclweb.org/anthology/P/P14/P14-1129.pdf.

Fraser, A. M. – Weller, M. – Cahill, A. – Cap, F. Modeling Inflection and Word-Formation in SMT. In *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*, p. 664–674, 2012. Available at: http://aclweb.org/anthology-new/E/E12/E12-1068.pdf.

Giménez, J. – Màrquez, L. Context-aware Discriminative Phrase Selection for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, p. 159–166, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. Available at: http://dl.acm.org/citation.cfm?id=1626355.1626374.

Gimpel, K. – Batra, D. – Dyer, C. – Shakhnarovich, G. – Tech, V. A Systematic Exploration of Diversity in Machine Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, p. 1100–1111, 2013.

Hajič, J. – Vidová-Hladká, B. Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. In *Proceedings of the COLING - ACL Conference*, p. 483–490, 1998.

Hajič, J. – Panevová, J. – Hajičová, E. – Sgall, P. – Pajas, P. – Štěpánek, J. – Havelka, J. – Mikulová, M. – Žabokrtský, Z. – Ševčíková-Razímová, M. – Urešová, Z. Prague Dependency Treebank 2.0, 2006.

Hajič, J. – Votrubec, J. – Krbec, P. – Květoň, P. – others. The best of two worlds: Cooperation of statistical and rule-based taggers for Czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies*, p. 67–74, 2007.

Hajič, J. – Hajičová, E. – Panevová, J. – Sgall, P. – Cinková, S. – Fučíková, E. – Mikulová, M. – Pajas, P. – Popelka, J. – Semecký, J. – Šindlerová, J. – Štěpánek, J. – Toman, J. – Urešová, Z. – Žabokrtský, Z. Prague Czech-English Dependency Treebank 2.0, 2011.

Heafield, K. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, p. 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. Available at: `http://www.aclweb.org/anthology/W11-2123.pdf`.

Hopkins, M. – May, J. Tuning as Ranking. In *EMNLP*, p. 1352–1362. ACL, 2011. Available at: `http://www.aclweb.org/anthology/D11-1125`. ISBN 978-1-937284-11-4.

Huck, M. – Tamchyna, A. – Bojar, O. – Fraser, A. Producing Unseen Morphological Variants in Statistical Machine Translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, p. 369–375, Stroudsburg, PA, USA, 2017. Universitat Politécnica de València, Association for Computational Linguistics. ISBN 978-1-945626-35-7.

Jeong, M. – Toutanova, K. – Suzuki, H. – Quirk, C. A Discriminative Lexicon Model for Complex Morphology. In *The Ninth Conference of the Association for*

Machine Translation in the Americas. Association for Computational Linguistics, November 2010. Available at: http://research.microsoft.com/apps/pubs/default.aspx?id=140709.

Johnson, H. – Martin, J. – Foster, G. – Kuhn, R. Improving Translation Quality by Discarding Most of the Phrasetable. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, p. 967–975. Available at: http://www.aclweb.org/anthology/D/D07/D07-1103.

Józefowicz, R. – Vinyals, O. – Schuster, M. – Shazeer, N. – Wu, Y. Exploring the Limits of Language Modeling. *CoRR*. 2016, abs/1602.02410. Available at: http://arxiv.org/abs/1602.02410.

Koehn, P. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, p. 79–86, Phuket, Thailand, 2005. AAMT, AAMT. Available at: http://mt-archive.info/MTS-2005-Koehn.pdf.

Koehn, P. *Statistical Machine Translation*. Cambridge University Press, 1st edition, 2010. ISBN 0521874157, 9780521874151.

Koehn, P. – Hoang, H. Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, p. 868–876, Prague, Czech Republic, June 2007. Association for Computational Linguistics. Available at: http://www.aclweb.org/anthology/D/D07/D07-1091.

Koehn, P. – Och, F. J. – Marcu, D. Statistical Phrase-based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, p. 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073462. Available at: http://dx.doi.org/10.3115/1073445.1073462.

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. – Herbst, E. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, p. 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. Available at: http://dl.acm.org/citation.cfm?id=1557769.1557821.

Li, Z. – Khudanpur, S. Large-scale Discriminative n-gram Language Models for Statistical Machine Translation. In *Proceedings of AMTA*, 2009. Available at: http://www.cs.jhu.edu/~{}zfli/pubs/discriminative_lm_for_smt_zhifei_amta_08.pdf.

Lin, C.-Y. – Och, F. J. ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation. In *Proceedings of Coling 2004*, p. 501–507, Geneva, Switzerland, Aug 23–Aug 27 2004. COLING. Available at: http://www.aclweb.org/anthology/C04-1072.

Marcus, M. P. – Marcinkiewicz, M. A. – Santorini, B. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.* June 1993, 19, 2, p. 313–330. ISSN 0891-2017. Available at: http://dl.acm.org/citation.cfm?id=972470.972475.

Mauser, A. – Hasan, S. – Ney, H. Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models. In *Conference on Empirical Methods in Natural Language Processing*, p. 210–218, Singapore, August 2009.

McDonald, R. – Pereira, F. – Ribarov, K. – Hajič, J. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, p. 523–530, 2005.

Niehues, J. – Herrmann, T. – Vogel, S. – Waibel, A. Wider Context by Using Bilingual Language Models in Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, p. 198–206, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. Available at: http://www.aclweb.org/anthology/W11-2124.pdf.

Och, F. J. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of ACL*, p. 160–167, Sapporo, Japan, 2003. ACL. Available at: http://aclweb.org/anthology-new/P/P03/#1000.

Papineni, K. – Roukos, S. – Ward, T. – Zhu, W.-J. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, p. 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. Available at: http://dx.doi.org/10.3115/1073083.1073135.

Peter, J.-T. et al. The QT21/HimL Combined Machine Translation System. In Bojar, O. – . (Ed.) *Proceedings of the First Conference on Machine Translation (WMT). Volume 2: Shared Task Papers*, 2, p. 344–355, Stroudsburg, PA, USA, 2016. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-10-4.

Popel, M. – Žabokrtský, Z. TectoMT: Modular NLP Framework. In Loftsson, H. – Rögnvaldsson, E. – Helgadottir, S. (Ed.) *IceTAL 2010*, 6233 / *Lecture Notes in Computer Science*, p. 293–304. Iceland Centre for Language Technology (ICLT), Springer, 2010.

Rosa, R. – Mareček, D. – Dušek, O. DEPFIX: A System for Automatic Correction of Czech MT Outputs. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, p. 362–368, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. Available at: `http://dl.acm.org/citation.cfm?id=2393015.2393066`.

Schmid, H. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing*, p. 44–49, Manchester, UK, 1994.

Sgall, P. *Generativní popis jazyka a česká deklinace*. Academia, 1967.

Shannon, C. E. A Mathematical Theory of Communication. *SIGMOBILE Mob. Comput. Commun. Rev.* January 2001, 5, 1, p. 3–55. ISSN 1559-1662. doi: 10.1145/584091.584093. Available at: `http://doi.acm.org/10.1145/584091.584093`.

Spoustová, J. – Spousta, M. A High-Quality Web Corpus of Czech. In Chair), N. C. C. – Choukri, K. – Declerck, T. – Doğan, M. U. – Maegaard, B. – Mariani, J. – Moreno, A. – Odijk, J. – Piperidis, S. (Ed.) *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Stolcke, A. SRILM – An Extensible Language Modeling Toolkit. In *Proceedings International Conference on Spoken Language Processing*, 2, p. 901–904, November 2002.

Straková, J. – Straka, M. – Hajič, J. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics:*

System Demonstrations, p. 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. Available at: http://www.aclweb.org/anthology/P/P14/P14-5003.pdf.

Subotin, M. An exponential translation model for target language morphology. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, p. 230–238, 2011. Available at: http://www.aclweb.org/anthology/P11-1024.

Sutskever, I. – Vinyals, O. – Le, Q. V. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, p. 3104–3112, Cambridge, MA, USA, 2014. MIT Press. Available at: http://dl.acm.org/citation.cfm?id=2969033.2969173.

Tamchyna, A. – Bojar, O. What a Transfer-Based System Brings to the Combination with PBMT. In Babych, B. – Eberle, K. – Lambert, P. – Rapp, R. – Banchs, R. – Costa-Jussà, M. (Ed.) *Proceedings of the Fourth Workshop on Hybrid Approaches to Translation (HyTra)*, p. 11–20, Stroudsburg, PA, USA, 2015. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-941643-67-9.

Tamchyna, A. – Braune, F. – Fraser, A. – Carpuat, M. – Daumé III, H. – Quirk, C. Integrating a Discriminative Classifier into Phrase-based and Hierarchical Decoding. *The Prague Bulletin of Mathematical Linguistics*. 2014a, 101, p. 29–41. ISSN 0032-6585.

Tamchyna, A. – Popel, M. – Rosa, R. – Bojar, O. CUNI in WMT14: Chimera Still Awaits Bellerophon. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, p. 195–200, Baltimore, MD, USA, 2014b. Association for Computational Linguistics. ISBN 978-1-941643-17-4.

Tamchyna, A. – Fraser, A. – Bojar, O. – Junczys-Dowmunt, M. Target-Side Context for Discriminative Models in Statistical Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1704–1714, Stroudsburg, PA, USA, 2016a. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-00-5.

Tamchyna, A. – Sudarikov, R. – Bojar, O. – Fraser, A. CUNI-LMU Submissions in WMT2016: Chimera Constrained and Beaten. In Bojar, O. – . (Ed.)

Proceedings of the First Conference on Machine Translation (WMT). Volume 2: Shared Task Papers, 2, p. 385–390, Stroudsburg, PA, USA, 2016b. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-10-4.

Tamchyna, A. Feature Selection for Factored Phrase-Based Machine Translation. Master's thesis, Charles University in Prague, 2012.

Tamchyna, A. – Bojar, O. No Free Lunch in Factored Phrase-Based Machine Translation. *Lecture Notes in Computer Science*. 2013, 7817, p. 210–223. ISSN 0302-9743.

Tiedemann, J. News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces. In Nicolov, N. – Bontcheva, K. – Angelova, G. – Mitkov, R. (Ed.) *Recent Advances in Natural Language Processing*, V. Borovets, Bulgaria: John Benjamins, Amsterdam/Philadelphia, 2009. p. 237–248. ISBN 978 90 272 4825 1.

Tufis, D. – Ion, R. – Ceausu, A. – Stefanescu, D. RACAI's Linguistic Web Services. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*, 2008. Available at: `http://www.lrec-conf.org/proceedings/lrec2008/summaries/90.html`.

Varga, D. – Németh, L. – Halácsy, P. – Kornai, A. – Trón, V. – Nagy, V. Parallel corpora for medium density languages. In *Proceedings of the Recent Advances in Natural Language Processing RANLP 2005*, p. 590–596, Borovets, Bulgaria, 2005.

Vickrey, D. – Biewald, L. – Teyssier, M. – Koller, D. Word-Sense Disambiguation for Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Vancouver, Canada, October 2005.

Wuebker, J. – Mauser, A. – Ney, H. Training Phrase Translation Models with Leaving-One-Out. In *Annual Meeting of the Association for Computational Linguistics*, p. 475–484, Uppsala, Sweden, July 2010.

Yarowsky, D. – Cucerzan, S. – Florian, R. – Schafer, C. – Wicentowski, R. The Johns Hopkins SENSEVAL2 System Descriptions. In *In Proceedings of SENSEVAL2*, p. 163–166, 2001.

Zeman, D. – Fishel, M. – Berka, J. – Bojar, O. Addicter: What Is Wrong with My Translations? *The Prague Bulletin of Mathematical Linguistics*. 2011, 96, p. 79–88.

Zens, R. *Phrase based statistical machine translation: models, search, training*. PhD thesis, 2008.