FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Martin Popel

Machine Translation Using Syntactic Analysis

Institute of Formal and Applied Linguistics

Supervisor of the doctoral thesis:	doc. Ing. Zdeněk Žabokrtský, Ph.D.
Study programme:	Informatics
Study branch:	Mathematical Linguistics

Prague 2018

Acknowledgements

First, I would like to thank my supervisor Zdeněk Žabokrtský for his excellent guidance through all the years of my studies, for many helpful and encouraging discussions and comments on the thesis and, last but not least, for his patience with me.

I am very grateful to my colleagues and friends at the Institute of Formal and Applied Linguistics, who supported me in many ways in my scientific work as well as numerous non-scientific activities. Your names are encoded in the word-embedding named-entity matrix floating somewhere nearby (unless my LaTeX decides otherwise in the final compilation).

I am obliged to the Google developers of the Tensor2Tensor framework for open-sourcing it and allowing me thus to base my research on a very strong baseline.

I thank Jana Straková, Markéta, my sister, and Jakub, her husband, for their invaluable comments on the draft of this thesis, including advices on grammar, style and structure of this thesis. Jakub is also responsible for explaining me how Oxford commas resolve ambiguities.

Finally, I would like to thank my parents, grandparents, Markétka, my wife, and Dorotka&Malvínka, our daughters. Your names are not embedded in the matrix, but in my heart. You have struggled with me in all times and senses, but still supported me with your love, time, cheers, when-it-will-be-finished talks, sandwiches, songs and tangled slinky toys. I hope I will have time for you now.

R	U	D	A	R	T	E	P	V	U	D	C	V	B	W
M	Q	F	D	L	U	C	I	E	H	லோ	க	நா	த	ன்
M	H	Q	E	V	A	L	U	D	N	E	V	R	V	N
O	N	D	R	A	L	E	Š	H	O	Š	I	M	O	Y
L	N	L	E	V	A	P	G	P	U	Š	G	C	J	L
M	A	R	K	É	T	A	K	Š	U	R	A	M	T	D
U	D	N	M	I	L	A	N	A	N	N	A	S	A	C
K	Z	A	R	D	N	I	J	A	R	E	A	I	D	T
N	L	Š	J	P	V	T	T	Y	A	Š	J	L	I	F
Z	A	U	A	A	K	A	T	K	A	M	A	V	V	K
D	D	D	N	M	L	O	A	J	Q	A	R	I	A	A
E	G	Š	A	K	Q	Z	M	O	T	R	K	E	D	R
N	A	V	A	O	N	Y	H	R	Q	T	A	R	C	R
Ě	M	N	V	O	R	O	M	A	N	I	X	A	I	Y
K	M	R	H	H	I	Z	X	X	K	N	F	R	Q	J

Figure 1: Word-embedding named-entity matrix.

In memory of my grandmother Miluška.

This work has been using language resources and tools developed and/or stored and/or distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

Title: Machine Translation Using Syntactic Analysis

Author: Martin Popel

Institute: Institute of Formal and Applied Linguistics

Supervisor: doc. Ing. Zdeněk Žabokrtský, Ph.D.,
Institute of Formal and Applied Linguistics

Abstract: This thesis describes our improvement of machine translation (MT), with a special focus on the English-Czech language pair, but using techniques applicable also to other languages. First, we present multiple improvements of the deep-syntactic system TectoMT. For instance, we implemented a novel context-sensitive translation model, comparing several machine learning approaches. We also adapted TectoMT to other domains and languages. Second, we present Transformer – a state-of-the-art end-to-end neural MT system. We analyzed in detail the effect of several training hyper-parameters. With our optimized training, the system outperformed the best result on the WMT2017 test set by +1.0 BLEU. We further extended this system by utilization of monolingual training data and by a new type of backtranslation (+2.8 BLEU compared to the baseline system). In addition, we leveraged domain adaptation and the effect of "translationese" (i.e which language in parallel data is the original and which is the translation) to optimize MT systems for original-language and translated-language data (gaining further +0.2 BLEU). Our improved neural MT system significantly ($p<0.05$) outperformed all other systems in English-Czech and Czech-English WMT2018 shared tasks, in terms of both automatic and manual evaluation. It was even significantly better than the human reference translation according to the manual evaluation.

Keywords: syntax-based machine translation, neural machine translation, deep neural networks, domain adaptation

Contents

Chapter 1

Introduction

For every kind of beasts, and of birds,
and of serpents, and of things in the sea,
is tamed, and hath been tamed of mankind:
But the tongue can no man tame;
it is an unruly evil, full of deadly poison.
James 3:7–8

1.1 Motivation

When I started my PhD studies I was fascinated by the intersection of linguistics, mathematics, machine learning and computer science, that is, by the broad area of computational linguistics and natural language processing (NLP). I chose machine translation (MT) as the main topic of my research for several reasons:

- MT is one of the most useful NLP applications in practice. Considerable efforts have been invested in its research in the past decades, including MT systems focusing on the English-Czech pair [e.g. Kirschner, 1974, 1982, Kirschner and Rosen, 1989, Čmejrek et al., 2003, Bojar, 2007, Žabokrtský et al., 2008]. However, due to the fascinating complexity of the task, there was still a lot of room for improvement in 2009.

- I already had some experience with TectoMT, a deep-syntactic MT system utilizing the tectogrammatical layer. Within my Master's thesis, I improved TectoMT's quality significantly using Hidden Markov Tree Model transfer [Popel, 2009]. My intuition was that the state of the art in 2009 – a phrase-based statistical machine translation (SMT) – is ignoring important properties of language and will eventually reach its limits even if trained on huge data sets unless it is complemented with methods that handle the hierarchical syntactic sentence structure. My hope was that improving TectoMT is the right way to go, as its architecture already models the source-language and target-language syntax and *only* needs to improve its three main phases – analysis, transfer and synthesis – with modern machine learning.

- I believed (and still believe) that developing high-quality MT will help us understand better how the language works. TectoMT as a linguistically motivated system seemed to be a great testbed for exploring the effect of individual components (taggers, parsers, translation models,...) and various dependency-tree attributes used as machine-learning features on the final translation quality. I enjoyed the opportunity to test linguistic theories and my intuition in practice (see a case study about clitics reordering in Section 3.2).

During my (rather long) research I've encountered several successes in improvements of the TectoMT system's quality (see Sections 3.3–3.5), as well as many attempts that seemed promising in restricted intrinsic evaluation, but proved unsuccessful when evaluated extrinsically in the complete translation task. I encountered also several unexpected challenges which in the end resulted in positive outcomes:

- In 2013–2016, I was involved in an European project called QTLeap,[1] the goal of which was a high-quality MT using deep language engineering approaches and its real-usage scenario was a multilingual helpdesk interface for IT troubleshooting. It has been decided that TectoMT will be adopted for English $\leftrightarrow$ Czech, Spanish, Portuguese, Dutch and Basque, which meant 9 new translation directions in addition to the existing English$\rightarrow$Czech. Many researchers were involved in QTLeap and there were several work packages, e.g. for lexical semantics modules, named entity linking, evaluation and web services. I was the main person responsible for the TectoMT translation in all language pairs and I implemented a draft version for English$\leftrightarrow$Spanish and English$\leftrightarrow$Portuguese. For this purpose, I redesigned TectoMT to be more easy to adopt for a new language pair and to follow guidelines suitable for a modern open-source software with many contributing developers (via GitHub[2]). QTLeap was very successful in several evaluations, automatic and manual, intrinsic and extrinsic, for all the mentioned language pairs except for English$\leftrightarrow$Basque [Gaudio et al., 2016].

- In addition to new language pairs, we had to deal with domain adaptation for the IT terminology and genre adaptation for the helpdesk questions and answers (our initial systems were lacking in translating questions and imperatives). Handling this properly was one of the keys to the success of QTLeap. This way (and also thanks to my involvement in the Khresmoi project [Pecina et al., 2014]), I became familiar with domain adaptation techniques and found this experience useful later on when working on Neural MT.

- I became interested in the manual evaluation of MT and co-authored an influential paper [Bojar et al., 2011] (with over 40 citations), which resulted in changes to the official evaluation method of WMT (Workshop on Statistical Machine Translation). I implemented a prototype of a toolkit for comparison and evaluation of machine translations, which was later reimplemented under name MT-ComparEval [Klejch et al., 2015, Sudarikov

[1] `http://qtleap.eu`

[2] `https://github.com/ufal/treex`

et al., 2016] by Ondřej Klejch within his bachelor's thesis under my supervision. MT-ComparEval and the related web services[3] were further improved within the QTLeap project and are popular among the users. I found the toolkit quite helpful myself when analyzing outputs of different versions of my MT systems as well as other systems.

- I become the main developer not only of TectoMT, but also of Treex – a Perl-based NLP framework in which TectoMT is implemented. Together with my supervisor Zdeněk Žabokrtský, I decided to redesign and reimplement the framework in Python and using Universal Dependencies[4] as the native representation and native file format (CoNLL-U). After thorough optimization, the new framework, called Udapi,[5] is about 60 times faster than Treex according to a benchmark [Popel et al., 2017]. In the end, I gave up the original plan to port TectoMT to Udapi, and instead provided tools for treebank developers, which were used in at least 20 treebanks from the Universal Dependencies and the number of users is still growing. While this achievement is not directly related to the thesis topic, it may be useful for syntax-based MT in future because Udapi is well-suited for exploring multi-lingual syntactically annotated data including parallel treebanks and for extracting features from treebanks (e.g. for NMT multitask learning).

- In 2011 and 2012, TectoMT was one of the winners of the constrained track of the WMT English→Czech news translation shared task (see Section 6.1). However, other systems improved much faster in the following years and the gap between the quality of TectoMT and the best system submitted to WMT became larger and larger, so in 2015–2016, TectoMT was one of the worst systems. My own attempts at combining TectoMT with phrase-based techniques were not successful. At the same time, my colleagues developed a system called Chimera combining phrase-based MT (Moses) with TectoMT and winning WMT in 2013–2015, outperforming even unconstrained systems such as Google Translate.

- Perhaps the biggest challenge in my research was the boom of Neural MT, which became the new state of the art in MT since 2016 (according to WMT results). I had to decide whether to finish my PhD thesis with the research I had done so far, but which was getting outdated, or whether to jump in to the new and fascinating area of deep learning and Neural MT, but starting almost from scratch. The final decision came in June 2017, shortly after a novel neural architecture called Transformer has been published [Vaswani et al., 2017]. I was impressed by the reported translation quality and even more by the (latent) structures emerging in

[3] `https://github.com/choko/MT-ComparEval`,
web services: `http://mt-compareval.ufal.cz/` and `http://wmt.ufal.cz/`

[4] `http://universaldependencies.org/`

[5] `http://udapi.github.io/`

self-attention layers, which had clear relations to syntax and semantics (see Section 2.5.3). Although the model was trained only on parallel sentences without any linguistic annotations, it learned to detect syntax-resembling sentence structures in a completely unsupervised way. I realized that it fits my thesis topic and that building my research on top of such a strong baseline would give the results much bigger impact and relevance. Within one year of very intensive experiments and research, I succeeded to improve the Transformer baseline significantly and surpass the state of the art according to the WMT 2018 shared task results.

In this preface about the genesis and motivations of this thesis, I use "I" to emphasize its subjective nature. In the rest of the thesis, I use authorial "we" for consistency as is usual in our field, but Appendix A explicitly distinguishes my own contributions.

1.2 Goals

The three main goals of this thesis are:

- Improve English↔Czech machine translation (MT) in such a way that could be easily applicable also to other language pairs.
- Explore the effect of syntactic structures in MT, both in a linguistically motivated system (TectoMT) with transfer on a deep-syntactic layer and in an end-to-end Neural MT system (Transformer) with latent syntactic structures.
- Explore domain adaptation techniques.

1.3 Outline

Chapter 2 reviews the related work in several areas related to the thesis. Chapter 3 presents our improvements of the TectoMT system. Chapter 4 reports our experiments with NMT training using the Transformer sequence-to-sequence model implemented in the Tensor2Tensor framework. Chapter 5 describes our research on the synergy of backtranslation and checkpoint averaging in NMT, including experiments which outperformed the state of the art. Chapter 6 provides a final evaluation and discussion. Finally, Chapter 7 summarizes the contribution of this thesis.

Chapter 2

Data and Related Work

Machine learning will be a phenomenal failure…
except in certain places, where its role will be crucial.
Martin Kay

In this chapter, we present first the English-Czech data sets (Section 2.1) and MT evaluation methods (Section 2.2) used in this thesis. Afterwards, we summarize different categories of MT system (Section 2.3) and describe two of the categories in more detail: tectogrammatical MT (Section 2.4) and neural MT (Section 2.5). Finally, we introduce the phenomenon of *translationese* (Section 2.6).

2.1 English-Czech Datasets

This section summarizes all the datasets we use for training, development evaluation (dev sets) and final evaluation (test sets) within the whole thesis.

2.1.1 Training Data

data set	sentence pairs (k)	words (k) EN	CS
CzEng 1.0	14 833	194 207	168 184
CzEng 1.7	57 065	618 424	543 184
Europarl v7	647	15 625	13 000
News Commentary v12	211	4 544	4 057
Common Crawl	162	3 349	2 927
QTLeap Batch1a	1	18	16

Table 2.1: Parallel training data sizes (in thousands).

Our primary training data comes from the training part of the CzEng parallel treebank.[1] CzEng is a collection of sentence-aligned English-Czech texts from European legislation, movie subtitles, fiction, parallel web pages, technical documentation and several smaller sources. All sentences are automatically parsed and analyzed up to the tectogrammatical layer using the Treex framework

[1] http://ufal.mff.cuni.cz/czeng

(see Section 2.4.1).[2] The word alignment induced using GIZA++ [Och and Ney, 2003] is projected up to the tectogrammatical layer. The English→Czech TectoMT results presented in this thesis (Chapter 3) were obtained by training on CzEng 1.0 [Bojar et al., 2012] and in case of domain-adaptation experiments (Section 3.7) also on the QTLeap Batch1a data set. Our NMT system (Chapters 4–5) was trained on CzEng 1.7 and three additional smaller sources of parallel data (Europarl, News Commentary, Common Crawl) downloaded from the WMT (Workshop on Statistical Machine Translation) website.[3] CzEng 1.7 is a subset of CzEng 1.6 [Bojar et al., 2016a] created by improved filtering.[4] In our NMT experiments, we use only the raw sentences from CzEng, without any annotation.

Table 2.1 reports the number of sentences and words (as computed using command `wc -w`) in our parallel training datasets. We use standard shortcuts M (for million) and k (for thousand) throughout the thesis. In the case of parallel data, we often use "*sentences*" as a shortcut for "*sentence pairs*".

data set	sentences (k)	words (k)	
		EN	CS
EN News Crawl 2016–2017	47 483	934 981	
CS News Crawl 2007–2017	65 383		927 348
– CS News Crawl 2007–2012	27 541		387 941
– CS News Crawl 2013–2015	25 054		357 855
– CS News Crawl 2016	5 636		79 809
– CS News Crawl 2017	7 152		101 743

Table 2.2: Monolingual training data sizes (in thousands).

In addition to the parallel training data, we also use monolingual News Crawl articles downloaded from the WMT website. For TectoMT experiments, we use Czech articles from years 2007–2012.[5] For NMT experiments in Chapter 5, we used different subsets of Czech articles from 2007–2017 and English articles from 2016–2017. See Table 2.2 for data sizes.

2.1.2 Development and Test Data

WMT shared task on news translation (see Section 2.2) provides a new test set (with approximately 3000 sentences) each year collected from recent news arti-

[2] CzEng 1.6 is available also in the CoNLL-U format with Universal Dependencies annotation, suitable for processing e.g. with the Udapi framework we developed [Popel et al., 2017].

[3] `http://www.statmt.org/wmt18` In 2016, WMT was renamed to *Conference on Machine Translation*, but keeping the legacy abbreviation WMT.

[4] `http://ufal.mff.cuni.cz/czeng/czeng17`

[5] We parsed this data in Treex and extracted a TreeLM (Section 2.4.3). For TectoMT Czech→English experiments, we extracted English TreeLM from the English side of CzEng 1.0.

data set	purpose	sentence pairs (k)	words (k) EN	CS
wmt08	dev	2.0	43	36
wmt13	dev/test	3.0	56	48
– wmt13-CZ	dev	0.5	10	8
– wmt13-nonCZ	dev	2.5	46	40
wmt08-16-CZ	dev	7.4	139	111
wmt08-16-nonCZ	dev	17.8	378	325
wmt17	test	3.0	55	47
wmt18	test	3.0	56	47
CzEng 1.0 dtest	test	151.3	1 986	1 719
QTLeap Batch2a	test	1.0	21	19
QTLeap Batch2q	test	1.0	10	8

Table 2.3: Evaluation data sizes (in thousands).

cles. The reference translations are created by professional translation agencies.[6] All of the translations are done directly, and not via an intermediate language. Test sets from previous years are allowed to be use as development data in WMT shared tasks.

For TectoMT (Chapter 3), we used `wmt08` (short name for WMT newstest2008) as our dev set and `wmt13` as our test set. In one experiment in Section 3.5.2, we used the "dtest" section of CzEng. In domain-adaptation experiments (Section 3.7), we used the QTLeap Batch2a test set. For NMT experiments (Chapters 4–5), we used `wmt13` as our dev set and `wmt17` as our test set. In an additional analysis in Section 5.3.12, we concatenated WMT test sets from 2008–2016 and divided them into `wmt08-16-CZ` and `wmt08-16-nonCZ` according to the country of origin. The final manual evaluation was done on `wmt18` (within the official WMT 2018 news task; see Chapter 6).

2.2 MT Evaluation

2.2.1 Manual Evaluation in WMT

We focus here on the manual (i.e. carried out by human annotators) evaluation done within the annual WMT shared tasks on news translation. To the best of our knowledge, WMT is the only MT shared task including the English-Czech pair. This pair is present in WMT each year since 2007. Both academic and commercial systems are regularly taking part in WMT. WMT evaluations are considered to be a benchmark for distinguishing state of the art in MT (as well as in MT evaluation methods) each year [Bojar et al., 2016b]. Within WMT, all submitted systems are evaluated both automatically and manually. However,

[6] For English-Czech: CEET in 2010–2012; Capita in 2013–2015; and Translated.net in 2016–2018.

only the manual evaluation is defined as the official measure of translation quality in WMT.[7]

WMT manual evaluation follows several general principles:

- The identity of systems is hidden to the annotators during evaluation.
- The final ranking of systems is accompanied by significance testing (see below).
- A detailed meta-evaluation is published each year [e.g. Bojar et al., 2016b, 2017a]. It includes statistics of inter-annotator agreement and also the number of human judgments per system (either pairwise comparisons or direct assessments, see below).[8]

The type of manual evaluation in WMT has changed several times. In 2008–2011 (with a pilot in 2007), the official manual evaluation was done by ranking up to five systems and computing the percentage of pair-wise comparisons where a given system was ranked as better or equal ($\geq$ranking). In 2012–2013, after showing drawbacks of this evaluation [Bojar et al., 2011], the official evaluation was changed to the percentage of non-tie comparisons where a given system was ranked as better ($>$ranking). In 2014–2016, the official evaluation was computed using the TrueSkill algorithm [Sakaguchi et al., 2014], allowing also negative scores.

In 2017, the evaluation was changed again, substituting the *relative ranking* of five systems by *direct assessment* (DA) [Graham et al., 2016], i.e. rating a single translation on a 0–100 scale [Bojar et al., 2017a]. Two types of final scores are reported: "Avg %" and "Avg z". "Avg %" is an average of all ratings from all annotators for a given system. In "Avg z", all ratings from each annotator are *standardized* first, i.e. scaled to have zero mean and standard deviation equal to one. **Statistical significance** is computed using Wilcoxon signed-rank (Mann-Whitney) test at p-level $p < 0.05$.

In 2018, in addition to the *reference-based* direct assessment (refDA, where the system translation is shown together with a reference translation), *source-based* direct assessment was also performed (srcDA, where the source-language sentence is shown instead of the reference). In srcDA, the human reference translation can be presented to the annotators in the same way as the evaluated MT systems. Thus, unlike refDA, the srcDA evaluation allows us to compare the quality of MT relative to the quality of human translation. The disadvantage of srcDA is that skilled bilingual annotators are required. For a discussion of further srcDA disadvantages, see Section 6.3.

[7] *"While automatic measures are an invaluable tool for the day-to-day development of machine translation systems, they are an imperfect substitute for human assessment of translation quality. Manual evaluation is time consuming and expensive to perform, so comprehensive comparisons of multiple systems are rare."* [Callison-Burch et al., 2007]

[8] This number for English$\rightarrow$Czech was 5590, 8554, 9624 and 2171 in WMT 2014–2017, respectively.

2.2.2 Automatic Evaluation

Most automatic metrics for MT evaluation are based on estimation of similarity between system translations and reference translations. Perhaps the most widely used metric is BLEU [Papineni et al., 2002]. BLEU is based on n-gram (unigrams up to 4-grams) precision of the translation relative to the reference translation and a brevity penalty to penalize too short translations [for details see Papineni et al., 2002].

There are several implementations of BLEU differing especially in the tokenization details. We use mostly the official case-insensitive (i.e. not distinguishing lowercase and uppercase letters) implementation `mteval-v14.pl`[9] with the `--international-tokenization` option because it has slightly better correlation with humans than the default ASCII-only tokenization [Macháček and Bojar, 2013].[10]

We report BLEU scaled to 0–100 as usual in most papers (although BLEU was originally defined as 0–1 by Papineni et al. [2002]); the higher BLEU value, the better translation. We measure confidence intervals and **statistical significance** using paired bootstrap resampling [Koehn, 2004] and consider a difference between two systems significant if $p < 0.05$.

There are several studies on reliability of BLEU and other automatic metrics [e.g. Callison-Burch et al., 2006, Macháček and Bojar, 2013, Bojar et al., 2017b]. The studies usually compute a correlation of the automatic metrics with various manual evaluations of MT quality. For example, there is a shared task on metrics evaluation within WMT (called *evaluation task* in 2008–2011 and *metrics task* in 2012–2018). On the one hand, BLEU is still the most popular MT metric and its correlation on system level (i.e. when computing BLEU on the whole test set, not on individual sentences) is relatively high, e.g. in English→Czech WMT2017, the Pearson correlation was 0.956 [Bojar et al., 2017b]. On the other hand, there are usually some better-correlating metrics (possibly different each year) and there are well known examples when BLEU may not correlate well [e.g. Callison-Burch et al., 2006], for example when the systems being compared are based on different paradigms, such as RBMT and SMT.[11] We evaluated all our results in Chapters 4–5 also with a character-based metric chrF [Popović, 2015], but the results were highly correlated with BLEU, so we omit these scores except for the final evaluation.

The automatic evaluation of MT becomes even more problematic when the quality of MT systems is similar to the quality of human references, or when it is even better. This affects not only BLEU, but all automatic metrics based

[9] `ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v14.pl`

[10] In Chapters 4–5, we use also `t2t-bleu` and the recent SacreBLEU (`https://github.com/awslabs/sockeye/tree/master/contrib/sacrebleu`) [Post, 2018] with options `-lc -tok intl`. Both these evaluation scripts produce results identical to the official implementation, but they are easier to work with. SacreBLEU automatically downloads the reference translation for a given WMT testset. It also reports a *signature* of the BLEU variant used, e.g. for the `wmt13` testset and options `-lc -tok intl`, the signature is `BLEU+case.lc+lang.en-cs+numrefs.1+smooth.exp+test.wmt13+tok.intl`.

[11] See e.g. Bojar et al. [2011] or footnote 4 on page 116.

on similarity with reference translations, especially if only a single reference is available (as is the case of WMT news translation task and several other shared tasks). Even if a system translation is judged better than a human reference, it does not mean that the system is better in all aspects (cf. Section 6.3). However, it is very difficult to distinguish which aspects of similarity with human references are the desired ones and which should be ignored instead.

For example, it seems that current state-of-the-art MT systems are (sometimes) better in specialized terminology (e.g. in economics, biology,. . .), where even fluent bilingual human translators may not choose the best translation unless they are experts in a given area. On the other hand, we can still see document-level inconsistencies caused by the fact that most MT systems translate each sentence independently. However, it is difficult to design a metric, which would take this into account; see e.g. Gong et al. [2015], Bawden et al. [2017] and papers submitted to the DiscoMT workshop.[12]

In conclusion, we still use BLEU (sometimes accompanied by other automatic metrics) within the development and evaluation of our systems. However, we are aware of its possible disadvantages and consider the manual evaluation more reliable.

2.3 Categories of MT Systems used in WMT

In this section, we provide a short overview of different MT system categories, with a special focus on the systems available for English→Czech MT in the last decade. Our selection is based on the systems submitted to the annual WMT shared tasks, which is used to define the state of the art; see Section 2.2.

We describe seven categories of MT systems: tectogrammatical MT, rule-base MT (RBMT), phrase-based statistical MT (SMT), syntax-based statistical MT, commercial online systems, hybrid systems and neural MT (NMT). Note that the categories are possibly overlapping (e.g. commercial online systems belong usually either to SMT or NMT; hybrid systems are by definition combining different categories). Our goal is not to define a perfect taxonomy of MT systems, but to provide a brief description of the systems. We use the categories defined here also in Section 6.1 when presenting paradigm shifts based on the WMT evaluation results.

Tectogrammatical MT

The Functional Generative Description (FGD) theory [Sgall, 1967] defines a so-called *tectogrammatical* layer of language description, where each sentence is encoded as a deep-syntactic dependency tree, which is supposed to represent the semantic structure of the sentence. See Section 2.4.1 for more details.

There were several MT prototype systems based on the tectogrammatical layer. The MAGENTA Czech→English system [Hajič et al., 2004] with a non-

[12] https://www.idiap.ch/workshop/DiscoMT/

isomorphic tree-to-tree transfer was based on Synchronous Tree Substitution Grammar [Eisner, 2003]. It was later improved by Čmejrek [2006] and Bojar [2008], who extended it also to English→Czech. An alternative tectogrammatical system of Čmejrek et al. [2003] used manually-written rules and automatically-extracted dictionary for generating English output directly from the tectogrammatical trees.

The most relevant for this thesis is a tectogrammatical system named TectoMT [Žabokrtský et al., 2008], developed in our institute since 2005 and participating in WMT since 2008 (see Section 2.4 and Chapter 3).

Rule-Based MT

We define RBMT as a category of fully rule-based MT systems, i.e. systems with manually written rules and no statistical training.[13] It should be noted that this category is very diverse – RBMT systems are based on various linguistic theories and representations. Well-known examples of RBMT systems are Systran [Toma, 1977] or Apertium [Forcada et al., 2011]. For English-Czech, there were early attempts using dependency syntax [Kirschner, 1982, Kirschner and Rosen, 1989], but the only two RBMT systems participating in English-Czech WMT are PC-Translator[14] and Eurotran XP.[15] Both are (or rather were) commercial closed-source software.

Phrase-Based Statistical MT

Almost all currently used MT systems (including Neural MT) are using statistics trained from data, but the term SMT is usually reserved for phrase-based systems, possibly including also "hierarchical phrase-based" systems [Chiang, 2005]. In this thesis, we follow the convention and define SMT as a category of phrase-based statistical MT, although we mention hierarchical systems within the category of Syntax-Based SMT.

Training of SMT systems consists of the following six steps:

- **Preprocessing** of the training data involves segmentation into sentences, one-to-one sentence alignment (source and target language), normalization (lowercasing or truecasing, simplifying typographical variants e.g. of quotation symbols), and tokenization to words.

- **Word alignment** means aligning each word in the source sentence to zero or more words in the target sentences. The standard approach (e.g. using the popular tool GIZA++ [Och and Ney, 2000]) is to induce independently source-to-target and target-to-source alignments (both one-to-many) using

[13] The TectoMT system mentioned in the previous subsection also uses rule-based components, but the most important parts of the translation pipeline (parsing and transfer models) are trained from data using machine learning techniques.

[14] `http://www.langsoft.cz/translator.htm`

[15] Eurotran XP website `www.eurotran.cz` is permanently unavailable.

IBM models [Brown et al., 1993], followed by symmetrization heuristics. IBM models were the basis of *word-based* SMT – a predecessor of phrase-based SMT.

- **Phrase extraction** means extraction of aligned phrase pairs (s, t) (source and target phrase) from the training data based on the word alignment. These phrases are overlapping (the maximal length of phrases is limited e.g. to 7 words).
- A **phrase table** is built from all the extracted phrase pairs. *Phrase table* is a list of unique phrase pairs, where each phrase pair is assigned several scores. The most important scores are forward and backward translation probabilities ($P(s|t)$ and $P(t|s)$), estimated by relative frequencies (and possibly smoothed). There are various methods for phrase-table filtering [Koehn, 2010].
- A **language model** (LM) is created, predicting $P_{LM}(f_i|f_1, ..., f_{i-1})$, where $f_1, ..., f_i$ is a sequence of words in the target language. The LM is estimated from target-language monolingual data, usually using an n-gram LM.
- A **log-linear model** is trained, usually using the MERT algorithm [Och, 2003].[16] The log-linear model assigns a weight λ_i to each feature $h_i(e, f)$, where e and f is the target- and source-language sentence, respectively. The number of features is usually lower than ten (MERT is not suitable for optimizing more features). The features include: a language-model score (based on P_{LM} for the whole target sentence); forward-translation and backward-translation scores (based on $P(t|s)$ and $P(s|t)$ probabilities stored in the phrase table); word and phrase penalty scores (number of words and phrases in the target sentence); and a distortion penalty (a measure penalizing reordering of phrases, i.e. changing the word order in translation).

Each of these steps is usually optimized independently of the other steps (in contrast to end-to-end training of NMT). During *decoding*, the target sentence is built left-to-right, using beam search to find a translation e that maximizes the log-linear model score $\sum_i \lambda_i h_i(e, f)$.

Perhaps the most popular SMT framework is Moses [Koehn et al., 2007]. The best English→Czech SMT system in WMT 2007–2018 was always based on Moses, although with different architectures and submitted by different teams (cf. Table 6.1). Further information about SMT can be found in *the* "SMT book" by Koehn [2010].

Syntax-Based Statistical MT

There is a subcategory of hierarchical phrased-based SMT systems [Chiang, 2005], where the phrases are allowed to contain "gaps", which will be translated

[16] The model is traditionally called *log-linear*, although its training in SMT does not involve proper estimation of probabilities and it would be more appropriate to call it a *linear* model.

by other phrases. Thus, each source sentence is represented as a phrase-structure tree, but in contrast to linguistically-motivated phrase-structure (constituency) trees, these trees have usually just a single non-terminal label. As in phrase-based SMT, a beam search is used to find the best segmentation into "hierarchical" phrases and the best-scoring translation.[17]

There are several other types of syntax-based SMT systems, generally divided into string-to-tree [e.g. Galley et al., 2004], tree-to-string [e.g. Huang et al., 2006], and tree-to-tree [e.g. Nesson et al., 2006, Eisner, 2003]. Some of these approaches use trees induced from the training data in an unsupervised way, some use dependency or constituency parsers trained on linguistically annotated data. Although a few of these systems were submitted to WMT in some years, they had always worse results than non-hierarchical SMT systems for English→Czech, and thus the category of syntax-based SMT is missing in our summary of WMT evaluations in Section 6.1.

Factored phrase-based SMT models [Koehn and Hoang, 2007] represent each word as a list of so-called factors, such as word form, lemma, part-of-speech tag or a word-class [Brown et al., 1992]. Including syntactic factors, e.g. a dependency relation label, is yet another way how to utilize syntax in SMT.

Online MT Systems

The category of online systems, such as Google Translate[18] or Bing Translator,[19] is special for several reasons: Many users are familiar with these systems and their quality. They cover a wide range of language pairs. Their training data is not constrained by the publicly available resources, so the systems are considered *unconstrained* in WMT. The details of the systems are not publicly known except for a small number of publications [e.g Wu et al., 2016] and press reports.[20] Based on available information, we know only that Google Translate started in 2006 as a phrase-based system, gradually improving and exploiting syntax for some language pairs, and since 2016 adopting end-to-end NMT.[21] Note that the names of online systems participating in WMT are traditionally anonymized (since WMT2010) as Online-A, Online-B etc.

Hybrid MT Systems (Chimera)

The category of *hybrid* systems is usually vaguely defined as systems combining techniques from multiple paradigms, most often RBMT and SMT. In our summary, we focus on a single hybrid system – Chimera [Bojar et al., 2013]. It is a system combining phrase-based Moses with deep-syntactic TectoMT and

[17] Due to search errors, the final translation found may not be the one with the highest score assigned by the model.

[18] https://translate.google.com/

[19] https://www.bing.com/Translator

[20] However, none of press reports describe any details about the Google Translate production system architecture.

[21] https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

also Depfix [Rosa et al., 2012] – a rule-based post-editing system for correction of grammatical errors.[22] First, TectoMT is used to translate the test set. Second, Moses is trained on both phrase tables – one with authentic translations from the parallel training data, and one extracted from the output of TectoMT on the test set. In an optional third step, Depfix is applied to the Moses translations.

Tamchyna and Bojar [2015] explore the impact of TectoMT on the translation quality of Chimera. They compare three systems: TectoMT, Moses and a system termed CH1, which is Chimera without Depfix, i.e. combination of Moses and TectoMT. In an experiment on the `wmt14` test set, 4.5% of tokens from the reference translation were present in CH1 and TectoMT, but not in Moses. Further 3.5% of reference tokens were present only in TectoMT (but not selected in the CH1 combination output). The CH1 combination was +1.65 BLEU better than Moses on `wmt14`. Applying Depfix on top of CH1 (thus creating the final Chimera configuration, which was the winner in WMT2014) gained further +0.2 BLEU improvement. After several other experiments (including manual evaluation), Tamchyna and Bojar [2015] conclude: *"We have found that [TectoMT] provides a mix of useful, correct translations and noise. Many of its translations are unavailable to the statistical component, so its generalization power is in fact essential. [...] Overall, we find that by adding [TectoMT] we obtain novel translations and improved morphological coherence. The final translation quality is improved significantly over both [Moses and TectoMT] alone, setting the state of the art for English→Czech translation for several years in a row."*

Neural MT

NMT is a category of end-to-end encoder-decoder-based Neural MT systems. See Section 2.5 for a description an overview of NMT.

The first NMT system [Jean et al., 2015] was submitted to WMT in 2015 by a team from the University of Montreal. In 2016–2017, the winning NMT system (and overall winner, also for other language pairs) was submitted by a team from the University of Edinburgh [Sennrich et al., 2017]. In 2018, the best English→Czech (and Czech→English) system is our NMT described in Chapters 4–5.

[22] In 2013–2016, the system [Bojar et al., 2013, Tamchyna et al., 2014, Bojar and Tamchyna, 2015, Tamchyna et al., 2016b] used all three components (Moses, TectoMT, Depfix) and the architecture was almost the same. In 2017, the system [Sudarikov et al., 2017] contained an NMT system in the combination.

2.4 TectoMT Overview

TectoMT is an MT system developed within a Perl-based NLP framework Treex at the Institute of Formal and Applied Linguistics (ÚFAL) since 2005. TectoMT is a very complex system and we cannot describe all its components in detail in this section. Instead, we present only a brief overview of the Treex framework and the three main translation phases – analysis, transfer and synthesis. More detailed description of selected improved components will be presented in Chapter 3.

2.4.1 Treex – NLP Framework

Originally, both the MT system and NLP framework were developed as one project under the name TectoMT [Žabokrtský et al., 2008, Popel and Žabokrtský, 2010]. In 2010–2011, the framework was redesigned, reimplemented and renamed to Treex. This was motivated by the fact that although the primary application of the framework was tectogrammatical MT, its general-purpose modular design proved to be useful in many other applications, not necessarily related to MT and tectogrammatics.[23]

Treex profits from the stratificational approach to the language, namely it defines four layers of language description (listed in the order of increasing level of abstraction): raw text (word layer), morphological layer, shallow-syntax layer (analytical layer, a-layer) and deep-syntax layer (layer of linguistic meaning, tectogrammatical layer, t-layer). The strategy is adopted from the Functional Generative Description (FGD) theory [Sgall, 1967, Sgall et al., 1986], which has been further elaborated and implemented in the Prague Dependency Treebank (PDT) [Hajič et al., 2006]. For technical reasons, Treex stores morphological layer and analytical layer together in a-trees – each a-node has both morphological attributes (form, lemma, tag) and shallow-syntax attributes (dependency relation to its parent).

Treex framework emphasizes modularity and reusability at various levels. Following the fundamental assumption that every non-trivial NLP task can be decomposed into a sequence of subsequent steps, these steps are implemented as reusable components called *blocks*. Each block has a well defined input and

[23] Treex/TectoMT has been used e.g. in machine translation based on Synchronous Tree Substitution Grammars and factored translation [Bojar and Hajič, 2008], aligning tectogrammatical structures of parallel Czech and English sentences [Mareček et al., 2008], building a large, automatically annotated parallel English-Czech treebank CzEng 0.9, 1.0 and 1.6 [Bojar and Žabokrtský, 2009, Bojar et al., 2012, 2016a], evaluating metrics for measuring translation quality [Kos and Bojar, 2009], complex pre-annotation of English tectogrammatical trees within the Prague Czech English Dependency Treebank project [Hajič et al., 2009], tagging the Czech data set for the CoNLL Shared Task [Hajič et al., 2009], gaining syntax-based features for prosody prediction [Romportl, 2008], experiments on information retrieval [Kravalová, 2009] and named entity recognition [Kravalová and Žabokrtský, 2009], conversion between different deep-syntactic representations of Russian sentences [Mareček and Kljueva, 2009] and a multi-lingual study on perplexity of n-gram and dependency language models [Popel and Mareček, 2010].

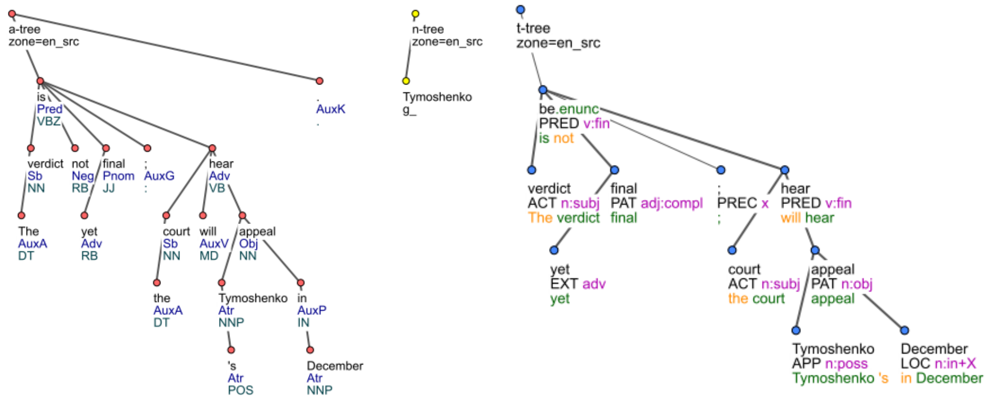


Figure 2.1: Source-language (English) representation of a sentence *"The verdict is not yet final: the court will hear Tymoshenko's applea in December."* Left: a-tree with with shallow dependencies and each node annotated with a word form, dependency relation ("afun") and PoS-tag (lemma is not shown). Middle: named entity tree. Right: t-tree with tectogrammatical lemmas, functors and formemes; word forms added only as a visual aid.

output specification and also a linguistically interpretable functionality in most cases, but there can be blocks for purely technical tasks (such as for feature extraction). This facilitates rapid development of new applications by simply listing the names of existing blocks to be applied to the data. Moreover, blocks in this sequence, which is called *scenario*, can be easily substituted with an alternative solution (other blocks or sub-scenarios), which attempts at solving the same subtask using a different approach. Scenarios and blocks can be adjusted also with parameters, e.g. for specifying which toolkit and model should be used for parsing.

For example, the whole English→Czech translation can be executed with the following command: `treex -Len -Ssrc Read::Sentences Scen::EN2CS Write::Sentences < in.txt > out.txt`,[24] where the main translation scenario is specified by `Scen::EN2CS`, which currently contains over 150 blocks (see Appendix B.2). In the following three sections, we summarize its three phases: analysis (of English sentences into English t-trees), transfer (of English t-trees into Czech t-trees) and synthesis (of Czech t-trees into Czech sentences).

2.4.2 TectoMT – Analysis

The input English text is segmented into sentences and tokenized into words. The words are tagged with Penn Treebank tags [Marcus et al., 1994] using the

[24] `treex` is the Treex command-line interface (executable). `-Len -Ssrc` specifies that the input is English and should be stored in a `src` *zone* (while the translations will be stored in a zone called `tst`). `Read::Sentences` is a so-called *reader* block, which specifies that the input is a plain text format, one sentence per line. Similarly, *writer* block `Write::Sentences` specifies the output format. For interactive translation with storing each sentence to a `*.treex` file (which can be visualized with the TrEd software [Pajas and Štěpánek, 2008]), one can use the following command `treex -Len -Ssrc Read::Sentences lines_per_doc=1 Scen::EN2CS Write::Treex Write::Sentences`.

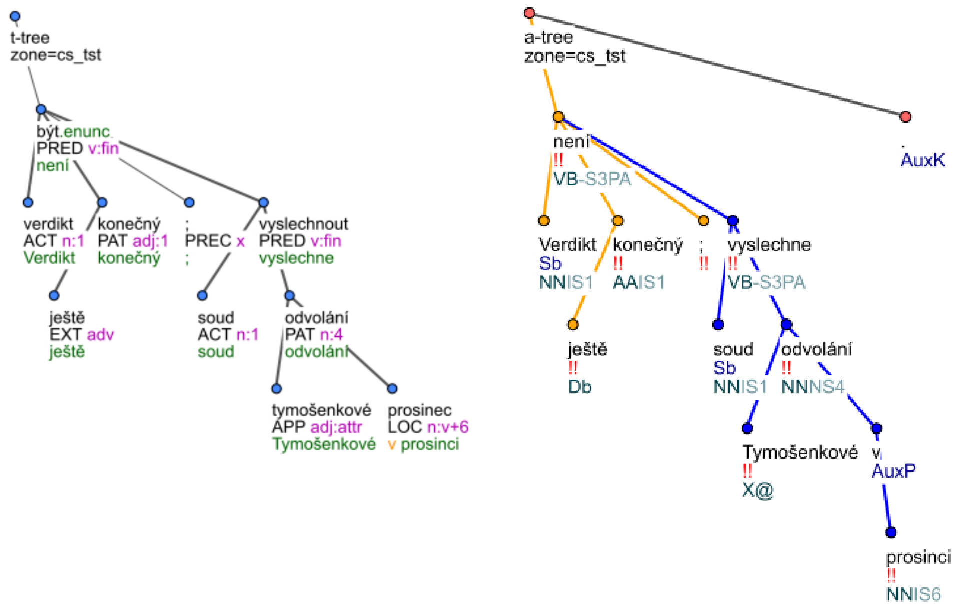


Figure 2.2: Target-language (Czech) representation of a translation of the sentence from Figure 2.1: "*Verdikt není ještě konečný: soud vyslechne Tymošenkové odvolání v prosinci.*" Left: t-tree. Right: a-tree. Dependency relation "afun" of some a-nodes is missing as it is not needed for the translation; the last line in each node shows a compact representation of Czech (PDT) morphological tags.

Morce tagger [Spoustová et al., 2007] and lemmatized using rules [Popel, 2009]. Subsequently, a dependency parser is applied (Maximum Spanning Tree parser [McDonald et al., 2005]) to create a shallow-syntax dependency tree (a-tree), where node corresponds to one word (see Figure 2.1 left).

We apply the NameTag named entity recognizer [Straková et al., 2014] and store the detected entities and their types in a named-entity tree (Figure 2.1 middle[25]).

Finally, each a-tree is converted to a tectogrammatical tree (t-tree, Figure 2.1 right) in a sequence of rule-based blocks.[26] Each *content* (*autosemantic*) word with its associated *functional* words is collapsed into a single tectogrammatical node (t-node), labeled with a lemma,[27] functor, formeme and semantically indispensable morphologically categories called *grammatemes* (e.g. number for nouns; tense and modality for verbs; gender, number and person for pronouns;

[25] Figure 2.1 originates from an older version of TectoMT, which used Stanford NER [Finkel et al., 2005] instead of NameTag and misclassified *Tymoshenko* as a geographical named entity (g_). NameTag detects it correctly as a personal name.

[26] Treex contains also machine-learning based blocks for detection of functors and for coreference resolution, but these are not used in the default translation scenario.

[27] In general, lemmas on the t-layer are different from the morphological lemmas. In English, we apply only two transformations: all personal pronouns are represented with a special lemma *#PersPron* and phrasal verbs are represented with a single t-node with a lemma joining the verb and particle, e.g. *make_up*. Tectogrammatical lemmas are usually called *t-lemmas*, but we don't consider the distinction important for our purposes and call them simply *lemmas*.

degree of comparison for adjectives and adverbs). Functors are semantic-role labels playing an important role in FGD and PDT [Sgall et al., 1986, Hajič et al., 2006], but less important in TectoMT (they distinguish coordination structures and they are used as features in translation models, but neither is crucial for the final quality). Formemes capture the morphosyntactic means which are used for expressing the tectogrammatical node in the surface sentence shape. Examples of formeme values are: `v:that+fin` – finite verb in a subordinated clause with conjunction *that*, `n:sb` – semantic noun in a subject position, `n:for+X` – semantic noun in a prepositional group with preposition *for*, `adj:attr` – semantic adjective in an attributive position. See Dušek et al. [2012] for details. Coreference links (both textual and grammatical) are introduced to connect anaphors with their antecedents. For this purpose, new t-nodes with lemma `#Cor` corresponding to unexpressed actors of infinitive verbs are created.[28]

2.4.3 TectoMT – Transfer

This section summarizes the baseline implementation of TectoMT transfer, while our improvements are described separately in Sections 3.3–3.4.

Transfer on the t-layer means transforming source-language (English) t-trees into target-language (Czech) t-trees and it is separated into five subtasks:

1. producing an n-best list of lemma translation variants,
2. producing an n-best list of formeme translation variants,
3. joint re-ranking of the n-best lists using Hidden Markov Tree Model (HMTM),
4. various ad-hoc blocks for addition or deletion of t-nodes or context-based correction of lemmas and formemes, and
5. translation of grammatemes and coreference.

Subtasks 1 and 2: Producing n-best lists of lemmas and formemes

A translation model is queried and the best-ranked (highest-probability) seven translation options for a given lemma (or a formeme) are stored in each node.[29] The baseline translation models are simple conditional probabilities $P(trg\_lemma|src\_lemma)$ and $P(trg\_formeme|src\_formeme, src\_parent\_lemma)$ estimated from

[28] We focus here mostly on the English→Czech translation direction, so we omit the description of Czech t-layer analysis (although it is needed for parsing of the parallel training data – CzEng). The Czech t-analysis principles are similar to English, with a few exceptions, e.g. new t-nodes need to be added also for dropped personal-pronoun subjects, which is frequent in Czech, but rare in English (except for imperatives and passives).

[29] We tried storing more than seven variants, but it didn't result in substantially better results and the decoding in subtask 3 was much slower. In addition, we limit the number of variants by their cumulative probability: 0.5 for lemmas and 0.9 for formemes. Again, we found these hyper-parameters as optimal on our development set.

relative counts in our training data – t-node aligned parallel treebank CzEng 1.0 [Bojar et al., 2012].[30] In Sections 3.3–3.4, we show more advanced and context-sensitive models.

Subtask 3: TreeLM reranking using HMTM

A globally optimal combination of lemmas and formemes is selected using HMTM [Popel and Žabokrtský, 2009] and a target-language tree model (TreeLM). HMTM is similar to standard (chain) Hidden Markov Model, but operates on trees instead of sequences. In HMTM, the *transition probability* defines how likely a node v (with hidden state variable $X(v)$) occurs given its parent $\rho(v)$ (with hidden state variable $X(\rho(v))$). Emissions correspond to nodes (with hidden states) emitting the observed output. The *emission probability* then describes how likely a hidden state $X(v)$ is to emit the observed output $Y(v)$. HMTM is well suited for describing generation/translation from syntactic trees – in particular, dependency trees are captured quite naturally by HMTM. In the context of deep MT transfer, observed variables correspond to labeled nodes in the source-language t-tree. The task is then to find the most probable assignment of hidden variables, which in turn correspond to labels of target-language t-nodes. A node label is a lemma-formeme pair. Transition probability is modeled by the TreeLM ($P(X(v)|X(\rho(v)))$), while emission probability is the probability of the particular source-language node being a translation of the hidden target-language lemma-formeme label ($P(Y(v)|X(v))$).[31]

This approach has an implicit assumption that the target-language t-tree has the same number of nodes and the same shape (topology) as the source-language t-tree, i.e. that the transfer is *isomorphic*. In practice, this assumption is not always true, but according to a manual evaluation [Popel, 2009, p. 14], only 8% of translation errors in the baseline were caused by this assumption. This supports the hypothesis that t-trees representing the same content in different languages are very similar,[32] while the amount of non-isomorphism in shallow-syntax trees is much higher (most sentences cannot be translated word by word).

[30] Even the baseline translation models were interpolated from several components, including rule-based dictionary of regular word-formative derivations, which can be helpful for translating some less frequent (but regularly derived) lemmas. For example, one of the derivation-based models estimates the probability $P(\textit{zajímavě}|\textit{interestingly})$ (a possibly unseen pair of deadjectival adverbs) by the value of $P(\textit{zajímavý}|\textit{interesting})$ (a frequent pair of adjectives).

[31] For example, the probability of the translation in Figure 2.2 according to HMTM is $P_e(\textit{be}|\textit{být}) \cdot P_t(\textit{být}|\textit{ROOT}) \cdot P_e(\textit{verdict}|\textit{verdikt}) \cdot P_t(\textit{verdikt}|\textit{být})$ and so on for all the nodes. For simplicity, we focus only on lemmas and exclude formemes in this example, but in reality, an important task of TreeLM is also to ensure the compatibility of lemma and formeme (within one node). We observed that in practice, we can obtain better results by using a *forward* translation model instead of the theoretically-correct *backward* model as the emission probability P_e, provided we tune a weight α in $P_e(\textit{být}|\textit{be})^\alpha \cdot P_t(\textit{být}|\textit{ROOT}) \cdot P_e(\textit{verdikt}|\textit{verdict})^\alpha \cdot P_t(\textit{verdikt}|\textit{být})$ etc. This way it is possible to exploit more source-language context in P_e, as described in Section 3.3.

[32] Or rather that the t-trees *can be* very similar, i.e. one of the acceptable translations has a t-tree with the same topology as the source t-tree.

It should be noted that the baseline system included several rule-based blocks to fix specific common types of non-isomorphism (and excluding these blocks would make the amount of errors caused by non-isomorphism higher than 8%).

Subtask 4: Lemma and formeme transfer postprocessing

Such correction blocks are included in subtask 4. For example, a t-node *rok* (*year*) is added when translating *in 1990* as *v roce 1990*. Some of these blocks are applied before subtask 3. For example, *make use of* can be translated as *použít, využít, používat* or *využívat*, and in all these cases two t-nodes (*make* and *use*) must be translated as one t-node. There is a special block with a list of such non-isomorphic translations. The advantage of applying this block first is that multiple translation variants can be specified and subtask 3 performs the final disambiguation, while considering the selected variant also for translation of the neighboring t-nodes. The disadvantage of this approach is that there is no way how a two-node translation could compete with a one-node translation (e.g. *prime minister* can be translated either as *předseda vlády* or *premiér*). Also, the list of non-isomorphic translations is collected manually and it is not integrated with the context-sensitive translation models discussed in Sections 3.3–3.4.[33]

Subtask 5: Transfer of grammatemes and coreference

Translation of grammatemes (e.g. tense and number) is much simpler than the translation of t-lemmas and formemes because these abstract linguistic categories are usually kept without any changes in the translation. Therefore, a set of relatively simple rules (with a list of exceptions) is sufficient for this task. For example, when translating *staff* as *zaměstanec* or *pracovník*, we change the grammateme `number` from singular to plural. Similarly to grammatemes, coreference links are usually kept unchanged, with a few exceptions when modifications are necessary. For example, we need to change textual coreference to grammatical (or create the links if they were missing) in case of possessive personal pronouns referring to a subject.[34]

[33] In addition to the discussed block (`T2T::EN2CS::TrLFPhrases`), there is another block (`T2T::EN2CS::OverridePpWithPhraseTr`) with a similar purpose, but it focuses on prepositional phrases only (most of which could be translated isomorphically, except for e.g. "*for Christ's sake*" $\rightarrow$ "*proboha*"), its list of translations was extracted automatically from the training data and only one translation option is provided (so the block is applied after subtask 3 and overrides its result).

[34] For example, "*He saw his house*" can be translated either as "*Viděl svůj dům*" or "*Viděl jeho dům*" depending on whether *He* and *his* refer to the same person or not. In English, the sentence is ambiguous and needs to be resolved based on context, thus the coreference is (by definition) *textual*. In Czech, the coreference of *svůj* is *grammatical* because *svůj* always refers to the subject (of a possibly non-finite verb) and the coreference can be resolved purely on the basis of grammatical rules.

2.4.4 TectoMT – Synthesis

In the synthesis phase, surface sentence is synthesized from the t-tree, which is basically the reverse operation of the tectogrammatical analysis. Figure 2.2 on page 29 shows a t-tree and a corresponding a-tree, which uniquely defines the surface sentence. The Czech synthesis consists of deleting personal pronoun subjects, adding punctuation and functional words (prepositions, subordinating conjunctions, auxiliary verbs, reflexive particles, expletive pronouns), spreading morphological categories according to grammatical agreement, arranging word order (see Section 3.2) and performing inflection. By inflection, we mean generating a word form for a given lemma and (possibly underspecified) morphological tag. We use an unigram morphological language model [Popel and Žabokrtský, 2009, p. 121] and Czech morphology database [Hajič, 2004] for this task.

2.5 Neural MT Overview

There were several early attempts to exploit *neural networks* (at that time also known as *connectionist models* or *continuous-space models*, in some contexts) in MT [e.g. Chrisman, 1991, Waibel et al., 1991, Forcada and Ñeco, 1997, Castaño et al., 1997]. However, the era of (modern) end-to-end NMT systems [Kalchbrenner and Blunsom, 2013, Sutskever et al., 2014, Bahdanau et al., 2014] started about two decades later, when the computational resources (GPU) became capable of training large models. NMT systems became competitive in well-known shared tasks: WMT 2015 [Jean et al., 2015], IWSLT 2015 [Luong and Manning, 2015] and WMT 2016 [Sennrich et al., 2016c] (cf. Section 6.1). Most of these systems use recurrent (RNN) layers (e.g. LSTM [Hochreiter and Schmidhuber, 1997] or GRU [Cho et al., 2014]), though some use convolutional layers as well [Kalchbrenner and Blunsom, 2013, Gehring et al., 2017]. Vaswani et al. [2017] introduced a novel model called Transformer, which uses *self-attention* instead of the recurrent or convolutional layers.

Transformer outperformed all the above-mentioned models [Vaswani et al., 2017] and it is used as our baseline system in Chapters 4–5. We thus describe the model in more detail (Section 2.5.2), after summarizing the basics principles of NMT architectures (Section 2.5.1). Finally, we explain what do we mean by the latent structures emerging in Transformer self-attention layers (Section 2.5.3).

2.5.1 General NMT Architecture

Token representation

In NMT, each input sentence is first tokenized into a sequence of tokens. The early NMT systems used words as the tokens [Sutskever et al., 2014], which resulted in large vocabularies and a necessity to handle unknown words. An alternative is to use characters as the tokens [Lee et al., 2016] or to use a hybrid word-character approach [Luong and Manning, 2016]. However, the most popular approach today is to split words into subword units (*subwords*)[35] and use these as the tokens for NMT [Sennrich et al., 2016b]. The subword vocabulary is trained so that frequent words are represented with a single subword, while rarer words are encoded into multiple subwords. There are several algorithms for training subword vocabularies [Schuster and Nakajima, 2012, Sennrich et al., 2016b, Macháček et al., 2018, Kudo, 2018].

Each token is represented as a real-valued vector, called *embedding* (word embedding, subword embedding or character embedding). These embeddings can be pre-trained on monolingual texts (e.g. with `word2vec` [Mikolov et al., 2013]), but most NMT systems initialize them randomly and train them jointly with the whole translation.

[35] For example, German word *Forschungsinstituten* (research institutes) is encoded with three subwords: *Forsch* + *ungsinstitu* + *ten_*.

Encoder-decoder architecture

Most NMT systems are based on an *encoder-decoder* architecture. The encoder maps the input sequence to a vector of *hidden states* (sometimes called *continuous representation* or *sentence embedding*). The decoder maps the hidden states into the output sequence (of target-language tokens). Each hidden state usually corresponds to one position (token) in the input sequence, so in general, the vector of hidden states has a variable length.[36] The early NMT systems [Sutskever et al., 2014] used only the last hidden vector as an input for the decoder. Thus, the training was forced to encode all the information about the input sentence into a fixed-length vector. Bahdanau et al. [2014] suggested to use a bidirectional GRU in the encoder. More importantly, they introduced an *encoder-decoder attention* mechanism, where the decoder has access to all encoder's hidden states. This way, when generating each output token, the decoder can *attend* to different parts of the input sentence. The encoder-decoder attention mechanism circumvents the fixed-length sentence-representation restriction and improves the translation quality, especially on longer sentences [Bahdanau et al., 2014].

Inference

The process of translating sentences (at test time) with a trained NMT model, is usually called *inference*.[37] Most NMT systems use *auto-regressive* inference.[38] This means that the output sentence is generated token by token and after each token is generated, its embedding is used as an input for generating the next token. In case of an RNN decoder, the decoding time grows linearly with the sentence length. The decoding finishes once the decoder generates a special end-of-sentence token.

Training

The advantage of NMT systems is that all their components can be trained in an end-to-end fashion. This is in contrast with SMT and TectoMT, where most components had to be trained separately. NMT is usually trained using back-propagation optimizing the *cross-entropy* loss of the last decoder's *softmax* layer, which predicts output token probabilities,[39] but there are also NMT systems

[36] For practical reasons of (mini-)batch training on GPU, sentences within one batch are usually padded to a fixed length according to the longest sentence in the batch (see Sections 4.3.4–4.3.5).

[37] In SMT, it is usually called *decoding*, but this is slightly ambiguous term in NMT because it can mean either the inference or only using the decoder (possibly in training time).

[38] There are also experimental systems with non-autoregressive inference [e.g. Gu et al., 2017, Zhang et al., 2018, Roy et al., 2018]. Their inference is faster, but the translation quality has not achieved the level of autoregressive models yet.

[39] $softmax(\mathbf{x})_j = exp(x_j)/\sum_i exp(x_i)$ and the *cross-entropy* loss is $H(\mathbf{y}, \mathbf{p}) = \sum_i y_i log(p_i)$, where $\mathbf{p} = softmax(\mathbf{x})$ is the predicted probability distribution of output tokens and $\mathbf{y}$ is the training-data "distribution", which is usually a *one-hot* vector ($y_k = 1$ for the token k on a given position in the reference translation, all other tokens $i \neq k$ have $y_i = 0$).

optimizing sentence-level metrics (e.g. BLEU or simulated human feedback) with reinforcement learning techniques [e.g. Nguyen et al., 2017].

Importantly, NMT usually uses the *teacher-forcing* technique: when generating the next word during training, it uses the previous word from the reference translation as the input instead of using the previously predicted word.

2.5.2 Transformer

Transformer [Vaswani et al., 2017] follows the general encoder-decoder architecture as described above, but instead of RNN layers it uses *self-attention* and feed-forward layers. This allows to speed up the training and partially also the decoding thanks to better usage of parallelism.[40]

Attention

In general form, *attention* can be defined as a function mapping three vectors of queries (Q), keys (K) and values (V) to an output vector, which is a weighted sum of the values V. The weight is computed as a compatibility of the corresponding key and query. In particular, Transformer uses a *scaled dot-product attention*:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.1}$$

where $Q \in \mathbb{R}^{n \times d_k}, K \in \mathbb{R}^{n \times d_k}, V \in \mathbb{R}^{n \times d_v}$, n is the sentence length, d_v is the dimension of values, and d_k is the dimension of the queries and keys.

In the *encoder-decoder attention*, keys and values come from the encoder's topmost layer and queries come from the decoder's previous layer.

In *self-attention*, all queries, keys and values come from the output of the previous layer. Self-attention is used both in encoder and decoder, but in decoder it is *masked*, so each position attends only to preceding positions because the following positions will not be known in inference time due to the autoregressive property of the decoder.

Multi-head attention

It is possible to use a single self-attention function in each layer, but the translation quality is improved [Vaswani et al., 2017] when combining multiple *attention heads*:

$$MultiHead(\hat{Q}, \hat{K}, \hat{V}) = \left[head_1(\hat{Q}, \hat{K}, \hat{V}), \ldots, head_h(\hat{Q}, \hat{K}, \hat{V})\right] W^O,$$

$$head_i(\hat{Q}, \hat{K}, \hat{V}) = Attention(\hat{Q}W_i^Q, \hat{K}W_i^K, \hat{V}W_i^V),$$

[40] During training both encoder and decoder work in a non-autoregressive mode, that is all positions are encoded/decoded in parallel. During inference, only the encoder works non-autoregressively.

where h is the number of heads; W_i^Q, W_i^K, W_i^V are parameter matrices which project original-size (d_{model}) queries, keys and values ($\hat{Q}, \hat{K}, \hat{V}$) into smaller-size vectors Q, K, V; and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ is a matrix which projects the concatenation of attention heads back to the original dimension d_{model}. Usually, the "smaller" dimensions d_v and d_k are set both to $\frac{d_{model}}{h}$, but other configurations are possible as well.

Encoder and decoder

The encoder of Transformer consists of 6 stacked layers of identical form:

$$layer(x) = LN\Big(x + PFFN\big(LN(x + MultiHead(x, x, x))\big)\Big),$$

where $x \in \mathbb{R}^{n \times d_{model}}$ is the input matrix; n is the input sequence length; LN is the *layer normalization* [Lei Ba et al., 2016]; $MultiHead$ is the multi-head self-attention sublayer described above; and *PFFN* is a position-wise feed-forward network (applied on each position independently, thus easy to parallelize):

$$PFFN([x_1, \ldots, x_n]) = [FFN(x_1), \ldots, FFN(x_n)]$$

$$FFN(x_i) = max(0,\ x_i W_1 + b_1)W_2 + b_2$$

The decoder is similar to the encoder, but in addition to the self-attention and feed-forward sublayers, each of the 6 layers includes also the encoder-decoder attention sublayer.

In the Transformer "BASE" model, $d_{model} = 512$, $h = 8$ and $d_k = d_v = 512/8 = 64$. In the "BIG" model, $d_{model} = 1024$, $h = 16$ and $d_k = d_v = 64$.

Positional encoding

Transformer contains no recurrence nor convolution and thus the information about position of the tokens in the sentence must be supplied by other means. Transformer encodes the absolute position ($pos \in \{1 \ldots n\}$) in the sequence into *positional encoding* vector $PE \in \mathbb{R}^{n \times d_{model}}$, which is subsequently summed with subword embeddings and provided as the input to the first layer of the encoder.

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

An alternative solution is to extend the self-attention formula with a term which depends on the relative distance of the key and query [Shaw et al., 2018].

Further reading

For more details on Transformer, see the original paper [Vaswani et al., 2017]. There is also a blog explaining Transformer with many illustrations, showing e.g. a visualization of the positional encoding.[41] Finally, Chapter 4 provides more information about training Transformer models.

2.5.3 Latent Structures in Transformer

The attention (Equation 2.1) is based on a "compatibility" function

$$softmax\left(\frac{QK^T}{\sqrt{d_k}}\right),$$

which assigns a weight $w_{i,j} \in \langle 0, 1\rangle$ to each query-key pair (Q_i, K_j). In the case of the multi-head self-attention in the encoder, the queries and keys are different projections of the vectors representing each token on a given layer. For a given sentence, encoder's layer and head, it is thus possible to visualize this self-attention as a weighted bipartite graph, where the weight of each edge is defined by $w_{i,j}$.

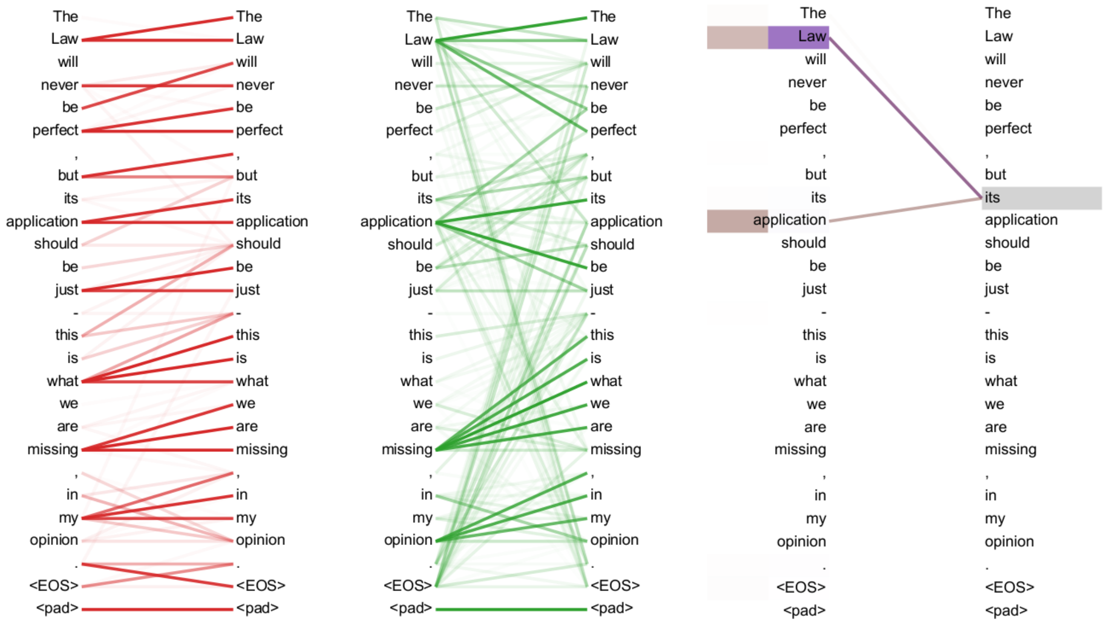

The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>

Figure 2.3: Visualization of self-attention in a Transformer model trained on English→German translation. Each of the three subfigures shows another attention head in encoder layer 5 (out of 6). The right-most subfigure shows two attention heads, but focusing only on the word '*its*' and illustrating coreference resolution. Incidentally, all words in the example sentence are frequent enough to be encoded with a single subword. Adapted from Vaswani et al. [2017].

Figure 2.3 shows an example of such visualization for different heads. Each head is visualized in a different color and edge weight is indicated by thickness. The words in the left column in each of the three visualizations represent vectors corresponding to these words on the input to the fifth layer of the encoder. By copying these vectors multiplied by the $w_{i,j}$ weights, Transformer starts building

[41] https://jalammar.github.io/illustrated-transformer/

the input for the sixth (and last) layer of the encoder. Thus we can imagine the words in the left column represent the sixth layer.[42]

We can see several interesting phenomena in Figure 2.3:

- The self-attention has a relatively sharp distribution – each position (in the right column) attends to a small number of positions (in the left column). Often, most of the attention focuses on a single position.
- Each head obviously specializes to a different task. The red-marked head in Figure 2.3 focuses mostly on short-distance dependencies and short phrases. The green-marked head focuses on longer-distance dependencies and longer phrases. The violet-marked head resolves the *its–Law* coreference link. Intuitively, this makes sense – by copying the vector representation of "*Law*" to the position of "*its*", we allow the further layers (in encoder and decoder) to "understand" the meaning of "*its application should be just*" and translate it correctly within a given context.
- The visualized self-attention structures resemble syntactic and semantic structures that are present in manually annotated treebanks. However, the self-attention structures emerged in the end-to-end training of English$\rightarrow$German translations, where no linguistic annotations were provided in the training data. We call these structures *latent* because they correspond to latent variables of the Transformer model, i.e. variables which are not (explicitly) visible in the data, but need to be inferred.

[42] As described in Section 2.5.2, self-attention is followed by the *PFFN* sublayer and both sublayers are wrapped by residual skip connections and layer normalization. However, self-attention is the only component, which combines representations on different positions.

2.6 Translationese

Translationese is a term used in translation studies (translatology). A text translated from language X into Y has different properties (lexical choice, syntactic structure, etc.) than a text originally written in language Y [Gellerstam, 1986, Koppel and Ordan, 2011]. Some scholars [e.g. Toury, 1995] consider the former (*X-specific translationese* of Y) a dialect of the latter (original Y). Some [e.g. Baker et al., 1993] emphasize general properties of *translationese*, which are independent of the source language X.

The main characteristics of translationese are:

- *Simplification*, which is "*the idea that translators subconsciously simplify the language or message or both.*" [Baker et al., 1993]
- *Normalization*, which is "*the tendency to conform to patterns and practices which are typical of the target language, even to the point of exaggerating them.*" [Baker et al., 1993]
- *Explicitation*, which is "*the process of introducing information into the target language which is present only implicitly in the source language, but which can be derived from the context or the situation.*" [Vinay and Darbelnet, 1958] (as cited by Baker [1998]) Explicitation is related to the *localization*, which includes cultural adaptation of the translation, e.g. explaining abbreviations or proper names which are common in the source language, but rare in the target language.

Explicitation is usually manifested by using more words in the translation, while simplification may result in using less words, although this interferes with the differing average number of words per sentence in different languages.

Translationese sometimes denotes rather negative aspects of translations which should be avoided by professional translators. Sometimes it denotes neutral or even positive aspects of translated language, for example localization or preserving ambiguities of the source language also in the target language if there is not enough context to resolve the ambiguity. Such ambiguity is usually a consequence of the source-language grammar, which may require (or prefer) to keep a given grammatical category unspecified (e.g. number of Chinese nouns, tense of Chinese or Indonesian verbs, definiteness of Czech nouns), while in the target language it would be more natural to express the grammatical category.

There are several works studying the effect of translationese on MT or utilizing it for tuning better MT systems; see Stymne [2017] for an overview.

Chapter 3

Improvements of TectoMT

Language data without trees is almost as arid as a desert.
Linguistic trees and grammars without data are decorative plants.
Data is the soil on which we should grow our trees.
But if we can't see the wood for the trees, it could be better to stay in the desert.
Khalil Sima'an

The baseline TectoMT system is described in Section 2.4. In this chapter, we describe selected improved components in greater depth.[1]

We start with a short evaluation of the influence of the dependency parser used in TectoMT on the final translation quality (Section 3.1). The second component is a rule-based block for reordering Czech clitics in the synthesis phase (Section 3.2). We selected this block as a case study, which should be representative also for other improved rule-based components, which are not discussed here. The other two selected components are alternative machine-learning-based context-sensitive translation models: MaxEnt (Section 3.3) and VowpalWabbit (Section 3.4). After evaluating these two models (Section 3.5), we describe how TectoMT has been adapted to new language pairs (Section 3.6) and domains (Section 3.7). We conclude with summarizing advantages and disadvantages of TectoMT (Section 3.8).

[1] This chapter is based on content published in Popel et al. [2011], Popel and Žabokrtský [2010], Mareček et al. [2010], Popel et al. [2015], Dušek et al. [2016].

3.1 Influence of Parsers

TectoMT allows to evaluate different NLP components in an extrinsic way. In Popel et al. [2011], we have analyzed the effect of five different parsers on the final translation quality. We observed that the correlation between intrinsic (parsing accuracy as measured by Unlabeled Attachment Score) and extrinsic (translation quality as measured by BLEU) evaluation is weak. This confirmed our hypothesis that what matters is not only the overall parsing accuracy, but also the parser-specific distribution of errors. We also observed that BLEU grows with the increasing amount of training dependency trees, but only at a sublogarithmic pace.

Finally, we developed SentChunk – a simple technique of dividing sentences into smaller chunks using parenthesis boundaries and parsing each chunk independently. This technique has almost no effect on the parsing accuracy, but improves the translation quality significantly (see Table 3.1).

			with SentChunk	
parser	UAS (%)	BLEU	ΔUAS (%)	ΔBLEU
MST [McDonald et al., 2005]	79.4	12.36	−0.1	+0.44
Malt [Nivre et al., 2007]	76.1	12.14	−0.1	+0.39
Zpar [Zhang and Nivre, 2011]	79.2	11.27	+0.1	+0.52
CJ [Charniak and Johnson, 2005]	90.4	12.84	–	–
Stanford [Klein and Manning, 2003]	82.5	12.77	–	–

Table 3.1: Influence of different English parsers on the final TectoMT translation quality. BLEU was evaluated on `wmt08`. UAS (unlabeled attachment score) was evaluated on Penn Treebank section 23. Dependency parsers (MST, Malt, Zpar) were trained on Penn Treebank (sections 2–21) converted to dependencies. Constituency parsers (CJ, Stanford) were trained directly on Penn Treebank and their output was converted to dependencies. All the reported ΔBLEU improvements caused by enhancing parsing with SentChunk are statistically significant ($p < 0.05$) according to paired bootstrap resampling [Koehn, 2004].

3.2 Reordering of Clitics

We choose the topic of reordering of Czech clitics as a case study to illustrate the abilities and dis/advantages of TectoMT relative to other MT approaches. There is a number of works describing the topic from the linguistic point of view, but we are not aware of any practical implementation apart from TectoMT.[2]

3.2.1 Czech Clitics Overview

A *clitic* is a morpheme that has syntactic characteristics of a word, but phonological characteristics of an affix.[3] While the word order of Czech is relatively free (encoding the Information Structure / Topic-Focus Articulation [Sgall et al., 1986]), the placement of clitics is usually restricted to the so-called *second position* in a clause, following the Wackernagel's Law [Wackernagel, 1892]. Similar rules hold for many other Indo-European (especially Slavic) languages.

For example, all the word orders in Example 1 are grammatical and the meaning is always the same except for the topic-focus differences. However, when we substitute the objects with pronoun clitics in Example 2, only word order 2a is grammatical. In 2c–2e, the clitics are not placed correctly on the second position. In 2b, the relative word order of the clitics is wrong.

(1) a. Dal Petrovi psa k Vánocům.
gave Peter$_{DAT}$ dog$_{ACC}$ for Christmas
'He gave Peter a dog for Christmas.'

b. Dal psa Petrovi k Vánocům.

c. Dal Petrovi k Vánocům psa.

d. Dal k Vánocům Petrovi psa.

e. Petrovi dal psa k Vánocům.

[2] Hana [2007] presents a formal description within *Higher Order Grammar* (HOG), but warns that "*HOG is not a programming language. The formalism is intended for linguists to describe and understand the problem, not for actually doing parsing, generation or assist in writing sms on a cell phone.*" He also points out: "*Although any six-year old native speaker* [can use sentences as Example 3] *without any problems, linguists have struggled for decades to uncover the principles that determine which orderings are possible.*" Admittedly, our goal in TectoMT is a bit easier – we need to find a single word order which is acceptable.

[3] Our linguistic description of clitics is very simplified for the purpose of brevity. We refer readers to Hana [2007, Chapter 4] and Rosen [2001, Chapter 7], who provide a detailed description with many examples (most examples in this section are taken from these two sources) and references to a number of other relevant works on clitics. Zwicky [1977] distinguishes *special* and *simple* clitics, where the example of the latter would be e.g. Czech monosyllabic prepositions, which have no accent of their own in the official Czech pronunciation, but their word-order position is fixed before the noun phrase, so the Wackernagel's Law does not apply on them. In this section, we focus only on the *special* clitics and call them for simplicity "clitics" as usual in other works.

(2) a. Dal mu ho k Vánocům.
gave him$_{DAT}$ him$_{ACC}$ for Christmas
'He gave it to him for Christmas.'

b. * Dal ho mu k Vánocům.

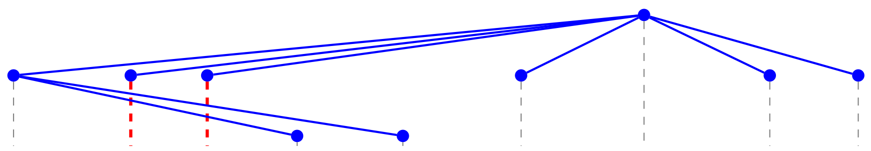
c. * Dal mu k Vánocům ho.

d. * Dal k Vánocům mu ho.

e. * Mu dal ho k Vánocům.

The sentence in Example 3 (visualized with its dependency tree) contains a complex verb phrase, where *opravit* is contextually bound and precedes the main verb *snažil* with a contextually unbound (in-focus) adverbial *marně*. The sentence contains a *clitic cluster* with four clitics (*jsem se mu to*), whose word order (one of the grammatical and common ones) results in a non-projectivity.[4]

(3)



Opravit	**jsem**	**se**	**mu**	**to**	včera	snažil	marně	.
to-repair	*aux*$_{1sg}$	*refl*	him$_{DAT}$	it$_{ACC}$	yesterday	tried	fruitlessly	.

'I tried to repair it for him yesterday without success.'

Non-projectivities are known to be difficult to analyze by parsers. When translating *into* Czech, we need to be able to *generate* non-projectivities, which causes problems to syntax-based MT (esp. when using constituency trees) and phrase-based SMT. A clitic may be placed far away from the verb it depends on (imagine *včera* is substituted by a long adverbial clause in Example 3). Also, this reordering is not optional but required by the grammar. Both these properties make the MT task even more difficult.

3.2.2 TectoMT Algorithm for Reordering Clitics

While some of the problems (e.g. wrong word order *ho mu* in Example 2b) can be prevented by an n-gram language model, this is not true in general and we have seen many cases of clitics-related errors in the SMT output (cf. evaluation results in Subsection 3.2.3). For example, note that all trigrams in 2d are grammatical, i.e. could be seen in grammatical Czech sentences, and should obtain high language-model scores. We need to use 4-grams to detect the ungrammatical[5]

[4] Words *jsem* and *se* are within the word-order span of the subtree governed by *Opravit*, but they do not belong to this subtree. As a result the edges (including the vertical dashed segments, highlighted in red) in the visualization are crossing.

[5] For some clitics and word orders, there is an unclear boundary between *ungrammatical* and *grammatical but uncommon*, as well as between what is acceptable by different native speakers [e.g. Hana, 2007, pp. 71, 117, 119, 132]. This boundary is not the focus of our work because TectoMT's task is to prevent both ungrammatical and uncommon word orders and ideally, select the most natural word order.

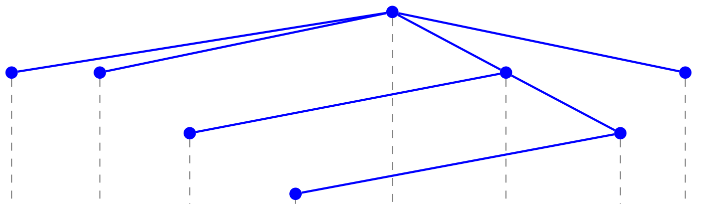
phrase *Dal k Vánocům mu*, but sometimes much longer n-grams may be needed (e.g. if substituting the adverbial *k Vánocům* by a longer phrase or clause).

Unlike SMT systems, TectoMT has access to the sentence dependency structure and morphological attributes of each word and can thus use explicit implementation of the linguistic rules underlying the clitics placement. There are four main steps of our implementation:[6]

1. Detect boundaries of clauses and process each clause independently.
2. Detect which words are clitics and should be moved.
3. Find the "second" position in a clause.
4. Move all clitics to that position in a correct relative order.

The first step is done in a separate TectoMT block as the clause segmentation is useful also for other purposes, e.g. insertion of commas. By *clauses*, we mean *finite clauses* governed by a finite verb. This is needed for the so-called *clitic climbing* shown in Example 4,[7] where clitics *mu* and *to* governed by non-finite verbs (*pomoct* and *najít*, respectively) are moved to the second position within the whole finite clause, not only within the respective non-finite verb phrase.

(4)

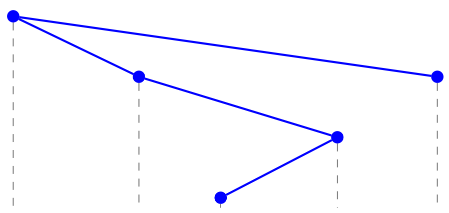


Všichni	**se**	**mu**	**to**	snažili	pomoct	najít	.
all	*refl*	him$_{DAT}$	it$_{ACC}$	tried	help$_{inf}$	find$_{inf}$	.

'Everybody tried to help him to find it.'

We need to restrict *clitic climbing* to verbal groups. In Example 5, only the word order 5a is correct – the clitic reordering is restricted to the non-finite clause *se vrátit* and cannot "climb over" the governing noun *důvod*, as shown in the ungrammatical word order 5b.

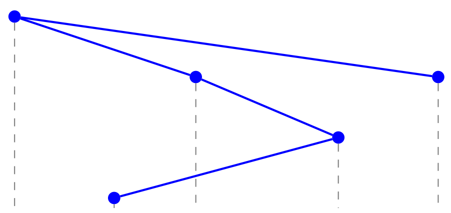
(5) a.



Nemá	důvod	**se**	vrátit	.
has-no$_{3sg}$	reason	*refl*	to-return$_{inf}$	.

'He has no reason for returning.'

b.



*Nemá	**se**	důvod	vrátit	.
has-no$_{3sg}$	*refl*	reason	to-return$_{inf}$	.

'He has no reason for returning.'

[6] https://github.com/ufal/treex/blob/master/lib/Treex/Block/T2A/CS/MoveCliticsToWackernagel.pm

[7] A projective word order with clitics moving only within their (possibly non-finite) verb phrases is acceptable here as well: *Všichni se snažili mu pomoct to najít.* However, forbidding the *clitic climbing* in all sentences results in unnatural word order and lower BLEU score.

The second step, detection of clitics to be moved, is relatively simple,[8] but we need to handle several exceptions. For example, Hana [2007, p. 123] discusses cases when a clitic cluster cannot contain two morphologically identical (*constant*) clitics, e.g. "**Kamila mi mi to slíbila vrátit*" (*Kamila promised me to return it to me*), where the first *mi* depends on *slíbila* (*promised*) and the second on *vrátit* (*to return*). In such cases, we need either to forbid *clitic climbing*, resulting in "*Kamila mi slíbila mi to vrátit*", or apply so-called *haplology* where only one of the identical words is retained, resulting in "*Kamila mi to slíbila vrátit*". We have observed also cases where only one of the two (possibly neighboring) identical words is a (*constant*) clitic. For example, in sentence "*Vláda je může snížit*" (*The government can lower them*) the clitic *je* (*them*) is moved to the second position. However, if we substitute *může* (*can*) with *je ochotna* (*is willing to*), where *je* is an auxiliary verb,[9] we would end up with "**Vláda je je ochotna snížit*", which is not acceptable in Czech. In this case, we cannot apply *haplology* and the only option is to forbid clitic climbing, resulting in "*Vláda je ochotna je snížit*". We have implemented this rule within the second step, by excluding such clitics from reordering.

The third step, finding the "second" position, basically means placing all the clitics after the last word of the first-position constituent (i.e. a dependency subtree). Again, there are several special cases to be handled. The first position may be a subordinating conjunction, which governs the verb in the Prague-style dependencies used in TectoMT. The first position may be the clause head (verb) itself. Heads of subordinate clauses and punctuation tokens have to be ignored when detecting the first position. Also some coordinating conjunctions (e.g. *a* and *ale*) have to be ignored [Rosen, 2001, p. 214], for example "*Přijď **nebo si to** najdi*" (*Come **or** find **it yourself***) versus "*Přijď **a** najdi **si to***" (*Come **and** find **it yourself***).

Finally, in the fourth step, we sort the clitics and move them to the second position. Our sorting is similar to Hana [2007, p. 121]: auxiliaries < reflexives < dative pronouns < accusative pronouns < rest < fringe (*tam, sem*). We take the clitics one by one in reversed order[10] and move them after the first-position

[8] We handle all *constant clitics* and some of the *inconstant clitics* [Rosen, 2001, p. 207] and *fringe clitics* [Hana, 2007, p. 90] (e.g. *tam*), where the reordering is optional – we cover the cases where reordering is more probable according to our `wmt08` development set. Depending on context, we include some instances of pronoun *to*, although it is not considered a clitic according to Rosen [2001, p. 212]. We exclude coordinated clitics, e.g. when substituting *mu* (*him*) in Example 2a with *nám i vám* (*to us and to you*), the correct word order is "*Dal ho k Vánocům nám i vám*" or "*Dal ho nám i vám k Vánocům.*", but not "**Dal nám i vám ho k Vánocům.*".

[9] Such auxiliary verb in periphrastic passive constructions is an *inconstant clitic* according to Rosen [2001, p. 210].

[10] Treex has an API support for moving one node (optionally with its subtree) to a given word-order position. However, it does not have any support for moving multiple nodes (which do not form a subtree) in one step. Thus, taking the clitics in reversed order was motivated by the following idea. E.g. when moving clitics A and B after the first-position subtree F, we first move B after F and then A after F (and thus before B). This works correctly, only if the clitics A and B are not part of the subtree F, in which case they end up in a reversed order. We realized this flaw only when writing this text and observing that instead of Example 2a, TectoMT produces incorrect 2b (*Dal ho mu k Vánocům*) because clitics A=*mu* and

	Moses	TectoMT
correct	56%	82%
not correct:		
– extra	22%	6%
– wrong word order	7%	4%
– other error	15%	8%

Table 3.2: Translation precision of the Czech clitic *se* for Moses and TectoMT.

subtree or after the clause-head node if it occupies the first position, or before the clause subtree if the first position is occupied by a subordinate conjunction (which governs the verb and is formally part of the preceding clause).

3.2.3 Manual Evaluation

Table 3.2 shows results of our manual evaluation focused on the clitic *se*, comparing phrase-based SMT system Moses and TectoMT translations. The evaluation was done (by the author of this thesis) on 200 sentences containing the clitic *se* selected from the `wmt14` test set. Each occurrence of the clitic *se* in the translation was classified either as correct or belonging to one of three categories of errors. "Extra" means that deleting the clitic would improve the translation (the governing verb was not a *reflexivum tantum* and did not need a reflexive pronoun). "Wrong word order" means that changing the position of the clitic in a sentence would improve the translation. "Other error" means that the governing verb was missing or the translation (only the part involving the clitic) was incomprehensible.

To complement this evaluation focused on precision, we carried out also a recall-focused evaluation. We selected 200 sentences where the reference contained word *se* and annotated the respective Moses and TectoMT translations. For each system, we counted cases when the clitic *se* was incorrectly missing and when it was used correctly. We ignored cases in which a given system chose an alternative translation with a verb not requiring the clitic (so the clitic was missing in the translation but it was not an error). Also in this recall-focused evaluation, TectoMT (15% incorrectly missing clitics) outperformed Moses (25%).

For other clitics than *se*, we have seen similar results (TectoMT better than Moses both in precision and recall), but our annotation was focused on sentences containing *se* and thus did not contain enough occurrences of other clitics for a proper evaluation.

B=*ho* depend on F=*Dal*. This could be easily fixed, by taking the clitics in non-reversed order and moving each clitic except the first one after the previously moved clitic. However, rather than fixing the error we document it as an example of a flaw which remained unnoticed over 7 years because it affects only a limited number of sentences.

3.2.4 Discussion

The goal of this case study was to show that TectoMT can handle intricate clitic word orders correctly, with which SMT struggles. The manual evaluation confirms that TectoMT is actually better in this aspect than SMT (Moses). Our implementation of the rules in TectoMT is mostly data-driven, i.e. we focus on fixing errors which are frequent in the translation output. Our implementation is missing treatment of several phenomena described in the literature,[11] but it handles also phenomena we have not found in the literature.[12]

We consider our rule-based linguistically-motivated approach as suitable and beneficial for a given goal.[13] The implementation is compact, consisting of about 250 lines of Perl code and comments. We were able to quickly test many hypotheses on real-world sentences – after each change of the rules, we translated the development set of 3000 sentences and checked the differences within few minutes (without any need to re-train machine learning models etc.). This allowed for fast development, prevention of regressions and also better understanding of the problematics.

Despite the advantages of the TectoMT approach and its good performance in the area of clitic reordering, it nevertheless has several limitations. First, the code is growing more complex and difficult to maintain over the years. Second, the code still does not handle correctly all edge cases (cf. footnote 10). Third, the rules can make more harm than good if the syntactic structure is wrong. For example, if we fail to recognize clause boundaries, a clitic may be moved too far away from its correct position. Keeping a clitic next to its governing verb (which is the initial position before applying the reordering block) may be ungrammatical, but it usually results in a comprehensible translation, while moving a clitic too far, possibly to a different clause may render the translation very difficult to understand.

[11] For example, we don't have any special rules for treatment of *ethical dative* [Hana, 2007, p. 116], where e.g. in "*On se ti vůbec nebál*" (*You know, he wasn't scared at all.*) the clitic *ti* (you_{DAT}) roughly corresponds to English phrase *you know*. However, even if a sentence with such a phrase is given as input to TectoMT, its (mostly isomorphic) transfer phase cannot currently produce such translation, so the absence of rules for ethical dative is not a limiting factor.

[12] For example, exclusion of coordinated clitics from the reordering and prevention of a *je je* phrase caused by clitic climbing.

[13] What we consider beneficial is the fact that TectoMT has access to the linguistically interpretable structures and that also the rules (reordering algorithm) is interpretable by humans. We can imagine an alternative implementation in TectoMT, where the reordering rules would be automatically learned from data using the linguistically-motivated features and structures we are using in our manually-written rules. Most of our conclusions would still apply.

3.3 MaxEnt Translation Model

Our lemma (and similarly for formemes[14]) translation models can be treated as probabilistic classifiers modeling $P(y|\mathbf{x})$, where y (output *label* or *class*) is the target-language lemma and $\mathbf{x}$ is a vector of input *features* from the source-language dependency tree. The most important feature is the lemma of the node being translated, but in this section and the following section (3.4), we describe two *discriminative* models, which use a much wider set of features.

The baseline translation models (subtasks 1–2 in Section 2.4.3) take only a very limited context $\mathbf{x}$ into account (and could be thus called *probabilistic dictionaries*). When translating a formeme, the probability predicted by the baseline TM is conditioned only by the formeme itself and its parent node's lemma (in the source tree). When translating a lemma, the probability is conditioned only by the lemma itself.[15] The translation probability is estimated simply based on a relative frequency. The baseline TM is *generative* because it could model also the joint probability $P(y, \mathbf{x})$. Adding more source-side features into such models is problematic because of data sparsity and because the features may be overlapping. However, high-quality translation is not possible when ignoring the context.[16]

In this section and the following section (3.4), we present two alternative approaches for building better translation models with more source-side context: a Maximum Entropy classifier and a VowpalWabbit classifier. We evaluate both in Section 3.5.

3.3.1 Rich-Context Source-Side Features

In this section, we use the following feature templates:

- lemma, formeme and selected morphological categories (grammatemes) of the given node,
- lemma, formeme and morphological categories of the governing node,
- lemmas and formemes of all child nodes,
- lemmas and formemes of the nearest linearly preceding and following nodes.

Figure 3.1 shows an example of these features in a sample sentence. Verb *cut* can be translated into Czech e.g. as *krájet, řezat, stříhat, zkrátit, kácet, sekat, snížit, smazat, oříznout, odstranit, zrušit* – English glosses are provided in the figure.

[14] See Section 2.4 for an overview of TectoMT, formemes, grammatemes and tectogrammatics.

[15] In the baseline models, lemmas include also PoS tags, e.g. `cut#V` for *cut* as a verb.

[16] In the baseline TectoMT translation, a limited target-side context was considered through HMTM (Section 2.4.3). However, sometimes a correct translation cannot be deduced from the neighboring words in the target tree (lemmas and formemes of the parent and children nodes) even if these are correctly translated. Moreover, a correct/acceptable translation may be missing in the 7 variants selected for HMTM reranking.

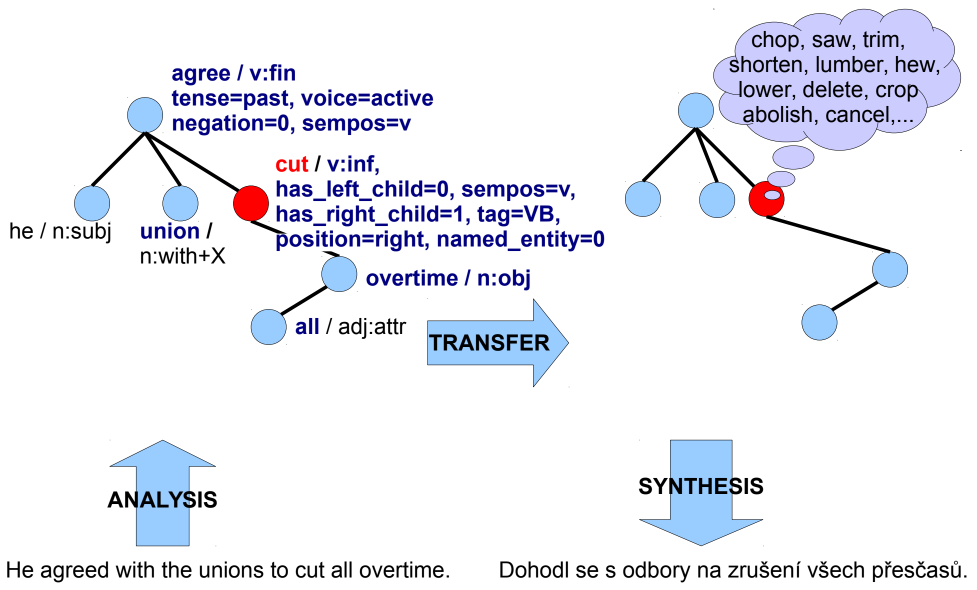

agree / v:fin
tense=past, voice=active
negation=0, sempos=v
cut / v:inf,
has_left_child=0, sempos=v,
has_right_child=1, tag=VB,
position=right, named_entity=0
he / n:subj
union / n:with+X
overtime / n:obj
all / adj:attr
TRANSFER
chop, saw, trim, shorten, lumber, hew, lower, delete, crop abolish, cancel,...
ANALYSIS
SYNTHESIS
He agreed with the unions to cut all overtime.
Dohodl se s odbory na zrušení všech přesčasů.

Figure 3.1: Illustration of our MaxEnt TM translating lemma *cut*. The features used in the model are marked in blue and bold font.

3.3.2 Implementation

A standard approach to probabilistic classification is a *multinomial logistic regression* also known as *maximum entropy classifier* (MaxEnt),[17] which has the following form

$$P_{MaxEnt}(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} exp \sum_i \lambda_i f_i(\mathbf{x}, y),$$

where f_i is a feature function, λ_i is its weight, and $Z(\mathbf{x})$ is the normalizing factor

$$Z(\mathbf{x}) = \sum_y exp \sum_i \lambda_i f_i(\mathbf{x}, y).$$

We use a Perl implementation `AI::MaxEntropy`[18] trained using the L-BFGS algorithm [Liu and Nocedal, 1989] batch training. We train a separate model for each source lemma which has at least 50 occurrences in our training data (there are 16 thousand such lemmas) and for time reasons we sample at most 10,000 occurrences (training examples). During translation, we interpolate the MaxEnt model with our baseline model 2:1 ($P(y|\mathbf{x}) = \frac{2}{3}P_{MaxEnt}(y|\mathbf{x}) + \frac{1}{3}P_{Base}(y|\mathbf{x})$), which was found on our development data as the optimal ratio. For lemmas with less than 50 occurrences, only the baseline model is used.

[17] In neural networks, it is known as a *softmax layer* (cf. Section 2.5.1). The term *maximum entropy classifier* is common especially in NLP [Berger et al., 1996, Ratnaparkhi, 1996], even when using regularization in the classifier and thus, strictly speaking, not optimizing towards a probabilistic distribution with the *largest entropy* (while satisfying given constraints). We use "MaxEnt" as an abbreviation for our particular implementation described here.

[18] `https://metacpan.org/pod/AI::MaxEntropy`

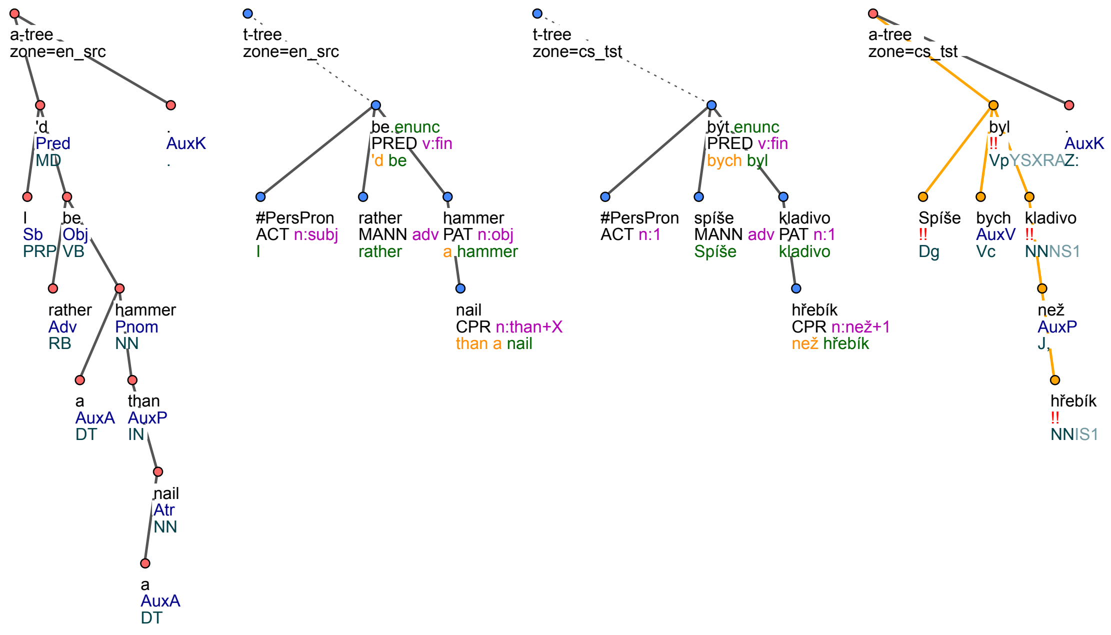


Figure 3.2: TectoMT translation of a sentence *"I'd rather be a hammer than a nail."* Left to right (following the translation process): English a-tree, English t-tree, Czech t-tree, Czech a-tree.

output_label=hřebík#N

feature	λ
child_formeme_n:in+X=1	1.64
is_member_of_coord=1	1.30
child_formeme_v:fin=1	1.04
next_lemma=down	0.84
is_capitalized=1	0.79
+precedes_parent=0	**0.75**
tense_g=post	0.74
+voice_g=active	**0.66**
prev_lemma=drive	0.66
parent_capitalized=1	0.62
formeme=n:from+X	0.60
+prev_lemma=hammer	**0.59**
child_lemma_few=1	0.55
child_lemma_remove=1	0.54
sempos=n.denot	0.50
next_lemma=and	0.50
formeme_g=v:until+fin	0.49
child_lemma_rusty=1	0.47
...	

output_label=nehet#N

feature	λ
child_formeme_n:poss=1	1.32
child_lemma_finger=1	1.07
child_formeme_n:of+X=1	0.98
precedes_parent=1	0.88
prev_lemma=black	0.77
child_lemma_broken=1	0.76
child_formeme_v:attr=1	0.70
formeme=n:at+X	0.67
formeme_g=n:attr	0.67
child_lemma_long=1	0.67
next_lemma=file	0.60
child_lemma_false=1	0.58
prev_lemma=false	0.58
+number=sg	**0.56**
formeme=n:obj	0.53
formeme=n:by+X	0.52
...	

Table 3.3: MaxEnt TM features and trained weights for translating lemma *nail* either as *hřebík* (*metal nail*) in the left table or as *nehet* (*fingernail*) in the right table. Features are sorted by their weights (λ) and only those with $\lambda > 0.45$ are shown. Features active in Figure 3.2 are marked with a + sign and bold font.

3.3.3 Interpreting Feature Weights

After training the MaxEnt model, there are ca. 4.5 million features with non-zero weight, out of which 2.4 million features are derived from the tree context (parent or child node), 1.1 million from the linear context (next or previous node) and 1.0 million from the node itself. This shows that the MaxEnt TM employs the dependency tree structure intensively (features which are not useful should get a zero weight thanks to the regularization used in training).

Figure 3.2 shows a translation of an example sentence *"I'd rather be a hammer than a nail."*, where the baseline model translates *nail* as *nehet* (*fingernail* or *toenail*), but the MaxEnt model chooses *hřebík* (*metal nail*). It is interesting to inspect the features indicative of each translation (Table 3.2). As expected, we can see features corresponding to phrases such as *nail in/down*, *drive a/the nail*, *rusty nail* when translating *nail* as *hřebík*, and *finger/black/broken/long/false nail*, *nail of*, *nail file* when translating *nail* as *nehet*. Note that the features `prev_lemma` and `next_lemma` take into account the preceding/following lemma according to the (surface) word order, but considering only nodes on the t-layer, thus excluding any prepositions, articles and other function words.

What may be a bit surprising is that being a member of a coordination (as a conjunct) is a feature indicative of *hřebík* (features `is_member_of_coord` and `next_lemma=and`, although the latter does not overlap with being a conjunct). Also *nail* following its parent node (feature `precedes_parent=0`) is indicative of *hřebík*, while *nail* preceding its parent is indicative of *nehet*. In English, a noun following its parent is typical for an object or a head of a prepositional phrase, but we can see than being a direct object (feature `formeme=n:obj`) is indicative of *nehet*, so we can conclude that it is rather the prepositional phrase what is indicative of *hřebík*. Similarly, a noun preceding its parent is typically a subject or a modifier, which is indicative of *nehet*, although the corresponding features (`formeme=n:subj` and `formeme=n:attr`) have weights < 0.45 and are thus not listed in the table.

We should be careful when interpreting the feature weights. By saying *"feature x_f is indicative of translation y_1"* (compared to an alternative translation y_2) when translating lemma l, we simply mean that $\lambda_l(x_f, y_1) > \lambda_l(x_f, y_2)$. It does not imply that $P(y_1|l, x_f) > P(y_2|l, x_f)$. For example, if l=*nail*, x_f=`'number=sg'`, y_1=*nehet* and y_2=*hřebík*, we can say that singular is indicative of *nehet*, but this does not necessarily mean that *nail* in singular (without knowing any other feature) is more probable to be translated as *nehet* rather than as *hřebík*. MaxEnt takes into account all the (possibly overlapping) features and it should hold that $\lambda_l(x_f, y_1) > \lambda_l(x_f, y_2) \Leftrightarrow P(y_1|l, \mathbf{x_f}) - P(y_2|l, \mathbf{x_f}) > P(y_1|l, \mathbf{x_{nf}}) - P(y_2|l, \mathbf{x_{nf}})$, where $\mathbf{x_f}$ is a set of all features, $x_f \in \mathbf{x_f}$ and $\mathbf{x_{nf}} = \mathbf{x_f} - \{x_f\}$.[19] In other words, if x_f is indicative of y_1 over y_2, the relative probability of y_1 compared to y_2 should be higher when x_f is present in the features.

[19] The biconditional is true if P is the probability predicted by the model. If the model is properly trained, it should reflect the training data probability, which may be different from the test data probability depending on the domain mismatch and amount of overfitting.

3.4 VowpalWabbit Translation Model

Although the MaxEnt TM is powerful, we have decided to compare it with a model trained with VowpalWabbit [Langford et al., 2007] machine learning toolkit.[20] We built one model for lemmas and another model for formemes. Here, we describe the lemma model; the formeme model is created in a similar way.

3.4.1 Advantages of VowpalWabbit over MaxEnt

- Only one model for all lemmas is trained instead of a separate model for each source lemma. This is technically easier to work with. It also allows exploring novel features shared across multiple source lemmas (so-called *multi-task learning* using *label-dependent features*; see below).

- The training is many times faster. Training MaxEnt lemma models on CzEng 1.0 takes more than one day when parallelized on 200 cores in a Sun Grid Engine cluster (one needs to wait until the last lemma model is trained). Training Vowpal Wabbit lemma model on CzEng 1.0 takes less than two hours (with 2-pass training) on a single machine (2 cores). Both approaches require to extract the training data into a suitable format, which can be easily parallelized and takes several hours on the 200 cores cluster. Therefore, Vowpal Wabbit allows researchers to try many more experimental setups than MaxEnt in the same amount of time.

- No pruning of training data is needed. In order to be able to train the MaxEnt models in reasonable time within 32 GiB memory, we had to limit the number of training instances per one source lemma to 10,000 and exclude source lemmas with less than 50 training instances. In VowpalWabbit no such pruning is needed because of the fast online learning and also because the model takes less space thanks to feature hashing [Ganchev and Dredze, 2008].[21]

- VowpalWabbit is trained using online learning, which allows domain adaptation using resumed learning (see Section 3.7).

3.4.2 One-Against-All Reduction in VowpalWabbit

Most machine-learning frameworks for multi-class classification expect that there is a fixed and relatively small set of classes (output labels) and the features provided by users are implicitly combined with all possible classes. In VowpalWabbit, we can train a multi-class classifier using a one-against-all (OAA,

[20] https://github.com/JohnLangford/vowpal_wabbit

[21] Moreover, the size of the model trained with VowpalWabbit can be adapted. We use 29-bit hash function, so the models take ca. 3 GiB of disk and 8 GiB of memory. By using 27 bits, we could scale down the model to 2 GiB of memory with only a tiny degradation in translation quality.

also known as one-vs-rest) reduction[22] to binary classification. For example, when translating lemma *nail*, we can assign integer labels to its possible translations: 1=*nehet#N*, 2=*hřebík#N*, 3=*přibít#V* (*finger nail*, *metal nail*, *to nail down*) and provide training examples in the following format:[23]

```
2 | prev_lemma=hammer voice_g=a precedes_p=0 num=sg
1 | prev_lemma=finger voice_g=a precedes_p=1 num=sg
```

This specifies two "positive" training examples (with correct translations 2=*hřebík#N* and 1=*nehet#N*, respectively). We list source-side features (following the vertical bar) in a sparse format, i.e. listing only the "activated" features (their default value is 1) and omitting the features not present for a given example. Each feature is implicitly combined with the output label. VowpalWabbit (within the OAA reduction) automatically generates "negative" examples for each of the positive examples, listing all "incorrect" translations and combining them with all features. Therefore, the underlying binary classifier is actually trained on these examples:

```
0 | 1_&_prev_lemma=hammer 1_&_voice_g=a 1_&_precedes_p=0 1_&_num=sg
1 | 2_&_prev_lemma=hammer 2_&_voice_g=a 2_&_precedes_p=0 2_&_num=sg
0 | 3_&_prev_lemma=hammer 3_&_voice_g=a 3_&_precedes_p=0 3_&_num=sg
1 | 1_&_prev_lemma=finger 1_&_voice_g=a 1_&_precedes_p=1 1_&_num=sg
0 | 2_&_prev_lemma=finger 2_&_voice_g=a 2_&_precedes_p=1 2_&_num=sg
0 | 3_&_prev_lemma=finger 3_&_voice_g=a 3_&_precedes_p=1 3_&_num=sg
```

Here the output labels 1 and 0 specify positive and negative training examples, respectively.

3.4.3 Single Model for All Lemmas

So far, we considered a model for the translation of a single lemma (*nail*) only. A naïve way how to extend it to a model for translation of all lemmas is to assign integer labels to all possible translations and enrich each feature with the source lemma, e.g. (assigning label 42 to *snížit#V*):

```
2 | nail_&_prev_lemma=hammer nail_&_voice_g=a nail_&_precedes_p=0 nail_&_num=sg
1 | nail_&_prev_lemma=finger nail_&_voice_g=a nail_&_precedes_p=1 nail_&_num=sg
42 | cut_&_prev_lemma=union cut_&_child_lemma=all
```

VowpalWabbit supports a compact representation of *interaction* features (i.e. quadratic, cubic,…) using so-called feature namespaces, so the previous sample can be simplified to:

[22] Reduction in machine learning means converting one task to another task. Usually a new/complex task is reduced to a known/simpler task, for which we already have a model.

[23] For simplicity, we restrict the toy example to only three possible translations and three features per line, but we follow the data format of VowpalWabbit. In practice, there are tens of features and up to 50 possible translations.

```
2  |L nail |S prev_lemma=hammer voice_g=a precedes_p=0 num=sg
1  |L nail |S prev_lemma=finger voice_g=a precedes_p=1 num=sg
42 |L cut  |S prev_lemma=union  child_lemma=all
```

Here, namespace `L` contains a single feature with the source lemma and namespace `S` contains all other source-side context features. If the training is executed with option `-q LS`, quadratic features combining the two namespaces will be created on the fly with the same effect as when explicitly enriching all features with the source lemma.[24]

However, this naïve OAA does not work in practice because we have over 2 million classes (all target-language lemmas), 112 thousand of them with at least 50 occurrences. Of course, when translating a given source lemma it does not make sense to consider translations that were never seen in the training data for this lemma. VowpalWabbit supports a *cost-sensitive one-against-all* reduction with *label-dependent* features, where the data format allows to list "possible" classes for each training example:

```
1:1 |T nail->nehet#N  |S prev_lemma=hammer voice_g=a precedes_p=0 num=sg
2:0 |T nail->hřebík#N |S prev_lemma=hammer voice_g=a precedes_p=0 num=sg
3:1 |T nail->přibít#V |S prev_lemma=hammer voice_g=a precedes_p=0 num=sg

1:0 |T cut->snížit#V  |S prev_lemma=union child_lemma=all
2:1 |T cut->krájet#V  |S prev_lemma=union child_lemma=all
```

Here, we encode the source lemma and target lemma in a single feature and place it in namespace `T` (and train with option `-q ST`). Each multi-class training example is specified on multiple lines, where each line represents a possible translation. Each line starts with an ID number and a *cost*, e.g. `2:0` means that the second translation option has cost=0, which means it is the correct translation (according to the reference). The ID number plays no role in the learning. While *cost-sensitive* OAA allows any real-valued costs (e.g. marking some translations as acceptable, but not ideal), we use always cost=0 for the correct translation and cost=1 for the rest.

3.4.4 Label-Dependent Features

The features in namespace `T` (e.g. `nail->nehet#N`) are termed *label-dependent* features because they depend on the output label (i.e. class, translation). In the previous sample, they actually define the whole label and they are the only way how the label is specified. However, we can also exploit label-dependent features that define the output label only partially, for example by specifying only the part-of-speech tag of the translation:

[24] Using `-q` is faster because of clever implementation of feature hashing in interaction features. When using `-q`, the "simple" features are retained, but in practice they do not affect the final predictions because these features are the same for all competing classes.

```
shared |S prev_lemma=hammer voice_g=a precedes_p=0 num=sg
1:1 |T nail->nehet#N  |P nail->#N
2:0 |T nail->hřebík#N |P nail->#N
3:1 |T nail->přibít#V |P nail->#V
```

Here, we add features specifying the translation's part-of-speech tag and source lemma into namespace `P`. We also use a compact format for specifying features (in namespace `S`) which are shared with all the lines in the multi-class training example.

When training without the features in namespace `P`[25] and with option `-q ST`, we use effectively the same set of features as in our MaxEnt model and we simulate a separate model for each source lemma. However, the `P` features cannot be included in our implementation of the MaxEnt model. These features are more robust (less sparse), e.g. the `nail->#N` feature is activated whenever *nail* is translated as a noun. These features can be even combined with the source-context features by training with option `-q SP`.

It is also possible to define features shared across multiple source lemmas, which is an example of *multi-task* learning when considering translation of each source lemma a separate task. For example, we can include features modeling only the part-of-speech-tag translation probability (e.g. `#V->#N`) or we can define features for translation of a whole group of lemmas with a similar meaning (e.g. `big/large/huge->velký#A`), where the correct translation may depend rather on the context (which will be modeled by interacting with features from the `S` namespace) than on the exact source lemma.

Our best setup according to the BLEU score includes also source formeme[26] in the `P`-namespace features but does not combine this namespace with all the features in namespace `S`:

```
shared |S prev_lemma=hammer voice_g=a precedes_p=0 num=sg
1:1 |T nail->nehet#N  |P nail_&_n-than+X->#N
2:0 |T nail->hřebík#N |P nail_&_n-than+X->#N
3:1 |T nail->přibít#V |P nail_&_n-than+X->#V
```

Concurrently with our work, Tamchyna et al. [2016a] used a similar model with label-dependent features in VowpalWabbit to improve an SMT system.

3.4.5 Predicting Probability Distribution

The cost-sensitive OAA with label-dependent features in VowpalWabbit predicts only the most probable class by default. However, in TectoMT we need a probability distribution for all the possible translations, so we can re-rank them with HMTM (and few other special-purpose re-ranking blocks). For this purpose, we have improved VowpalWabbit by implementing the option `--probabilities`,

[25] Either by excluding these features from the training data, or by specifying option `--ignore P`.

[26] VowpalWabbit uses colon for separating feature name from its (optional) value. We thus need to escape colons in feature names and encode formeme *n:than+X* e.g. as `n-than+X`.

which should be used together with `--loss_function=logistic`, because the default squared loss function does not result in proper probabilistic distribution.[27] The option `--probabilities` also instructs VowpalWabbit to report the multi-class logistic loss, which we consider a better intrinsic quality indicator for our purposes than the zero-one loss which is reported by default. The exact command we use for training is:

```
vw -d train-data.gz -f final.model -c --holdout_off -l 3 --passes=2 \
   --loss_function=logistic --csoaa_ldf=mc --probabilities -b 29 -qST
```

3.4.6 Using VowpalWabbit for Word Sense Disambiguation

We have also used VowpalWabbit in other NLP tasks, successfully utilizing the approach we developed originally for MT. For example, in Dušek et al. [2015a] we report on improving the state of the art in Word Sense Disambiguation of verbs (namely, valency frame detection) on the Prague Czech-English Dependency Treebank (PCEDT) [Hajič et al., 2012]. We improved labeled F1 score from 80.30% to 82.39% only by using VowpalWabbit instead of LibLINEAR logistic regression [Fan et al., 2008], with the same set of features. VowpalWabbit with label-dependent features allowed to use also features from bi-lingual alignment and valency lexicon, which further improved the F1 score to 82.93%. Prediction of valency frames based on dependency-tree-based features is a similar task to predicting target-language translations, but easier because the set of possible valency frames is (almost always) smaller than the set of possible translations and because the valency frame detection is the final task in this experiment (unlike lemma translation in TectoMT, which is followed by HMTM reranking and the synthesis phase).

[27] In our experiments, optimizing for the squared loss and reducing the classification to a regression problem leads to the best accuracy (considering only the top-ranked translation). However, the best Recall@6 (see Section 3.5) and best BLEU is achieved when optimizing for the logistic loss and reducing the classification to binary classification (logistic regression).

3.5 Evaluation of Translation Models

3.5.1 Intrinsic Evaluation

First, we compare our TMs in a study focused on translation of a single lemma *cut*.[28] We call this *intrinsic* evaluation because we evaluate only the task solved by the models, not our final task – translation of whole sentences (or documents).

The training and test data for this study were sampled from the CzEng treebank. The training data consist of 5901 training examples with 1293 possible translations, out of which 865 have only a single occurrence (most of these are caused by alignment errors in CzEng). The test data consist of 758 examples, out of which only 639 have a translation which is contained in the training data. Thus the "oracle" accuracy is 639/758=84%; no classifier treating lemmas as atomic units can have a higher accuracy and predict translations not seen in the training data. The baseline accuracy of 7% corresponds to predicting always the most frequent translation of *cut*, which is *vyjmout* (*take out / remove / pull out*) and 90 out of the total 1293 test examples use this translation. We report also recall@N (the percentage of cases when the correct translation is among the N best ones according to a given model's ranking), motivated by the fact that HMTM considers the N-best list predicted by TMs.

model	Accuracy (%)	Recall@6 (%)
oracle (seen in train)	84	84
Baseline TM	7	25
MaxEnt TM	10	12
NeuralNet TM	33	50
VowpalWabbit TM	35	51

Table 3.4: Intrinsic evaluation of translation accuracy for lemma *cut*. Recall@N is the percentage of cases when the correct translation is among the N best ones according to a given model's ranking. Accuracy corresponds to Recall@1.

Table 3.4 presents results of our intrinsic evaluation. Baseline TM is the model based only on relative frequency of lemmas (without any context). NeuralNet TM is a feed-forward neural network with a single hidden layer with 600 neurons.[29] MaxEnt's accuracy is only 3% higher than the baseline. Surprisingly,

[28] We chose the lemma *cut* because it is difficult to translate without any context (it is one of the top 100 English words with the highest entropy of $P(cs|en)$ according to CzEng alignments). When evaluating MaxEnt on all lemmas [Mareček et al., 2010], its accuracy was 57% (vs. 10% for *cut*) because there are many easy-to-translate lemmas. We used *cut* and several other lemmas when developing prototypes of our models. Some of the prototypes, namely NeuralNet TM, were never trained for all lemmas (and evaluated within TectoMT), thus we resort to *cut* in this study and Table 3.4.

[29] This unpublished research on using neural networks for translation was done in 2012 in collaboration with Markéta Tomková and Jakub Tomek. While the results were promising, the networks had huge memory and disk-space requirements, so we abandoned this line of research and focused on VowpalWabbit, which uses feature hashing to limit the memory

TectoMT version	BLEU	
	No LM	TreeLM
Baseline TM	10.70	12.15
MaxEnt TM	12.78 (+2.08)	13.57 (+1.42)
VowpalWabbit TM	13.07 (+2.37)	13.77 (+1.62)

Table 3.5: BLEU evaluation of our two context-sensitive models, with and without a TreeLM (target-language tree model applied via HMTM) on the wmt13 test set.

its recall@6 is 13% worse than the baseline.[30] This was one of our motivations for switching to a better learning algorithm – VowpalWabbit, whose accuracy is substantially better than MaxEnt: +25% (and recall@6 is also better: +39%).

We did a similar evaluation for a broader set of lemmas and while the absolute scores differed, the relative ordering of our systems was always the same (VowpalWabbit being much better than MaxEnt, which was slightly better than the baseline).

3.5.2 BLEU Evaluation

According to the BLEU evaluation on the wmt13 testset (Table 3.5), VowpalWabbit is significantly ($p < 0.05$) better than MaxEnt and MaxEnt significantly ($p < 0.05$) outperforms the baseline TM. Both MaxEnt and VowpalWabbit benefit from re-ranking by TreeLM (via HMTM; see Section 2.4.3), although its relative improvement (+0.79 for MaxEnt and +0.70 for VowpalWabbit) is lower than the improvement of the baseline TM (+1.45). TreeLM is trained on big target-language monolingual data, but the effect of target-side context-sensitivity of TreeLM is partially overlapping with the source-side context-sensitivity of MaxEnt and VowpalWabbit.

The improvement of VowpalWabbit relative to MaxEnt is only +0.20 BLEU, which is much less than what we hoped for based on the intrinsic evaluation. One possible explanation is that VowpalWabbit overfitted the CzEng "domain", which is not optimal for WMT test sets.[31]

Note that in our domain-adaptation setting, the improvement caused by using VowpalWabbit instead of MaxEnt is much bigger (+1.57 BLEU), see Section 3.7.

requirements. Later we tried adding a hidden layer to VowpalWabbit, but without any improvement in BLEU. We should emphasize that in these approaches, neural networks were used only for the transfer phase (not in the end-to-end manner as in Chapters 4–5).

[30] In TectoMT, we interpolate MaxEnt with the baseline TM, which partially compensates this.

[31] VowpalWabbit achieved substantial improvements over MaxEnt when evaluated on the CzEng *test* data set (dtest), so this is not the typical case of overfitting to train vs. test data. We have also tuned the number of training epochs (passes) in VowpalWabbit using WMT development set to prevent this kind of overfitting.

3.6 New Language Pairs in TectoMT

Until 2014, TectoMT was available only for the English→Czech translation direction, although it had always been hoped that more language pairs will be included in the future. That moment came in 2014–2016, when TectoMT was extended to four more languages: Spanish, Portuguese, Dutch and Basque (es, pt, nl, eu) within the QTLeap project.[32] Consequently, there are ten translation directions available in TectoMT: en↔cs, en↔es, en↔pt, en↔nl, en↔eu.[33]

In order to facilitate language-independent solutions, we decided to use two projects carried out at ÚFAL: Interset and HamleDT. Interset [Zeman, 2008] is a set of morphological (and some syntactic) features with a mapping to many languages and morphological tag sets. HamleDT [Zeman et al., 2012, 2014, Rosa et al., 2014] is a collection of dependency treebanks (for 36 languages in the latest version HamleDT 3.0), defining a common annotation style based on the Prague Dependency Treebank [Hajič et al., 2006]. We used and improved the software developed originally within these two projects, which allowed us to reuse existing taggers and parsers for the four new languages and convert the morphological and syntactic annotation to a common Interset+HamleDT style.

Most TectoMT rules for the conversion from a-layer to t-layer have been adapted to expect Interset morphological features and HamleDT-style dependencies, which improves their usability for different languages. The implementation involves a common base classes with language-specific subclasses, e.g. `A2T::ES::SetFormeme` is a subclass of `A2T::SetFormeme`. Similarly, we have created extensible base classes for common tasks in the transfer and synthesis phase. This greatly simplified the adoption of the new language pairs into TectoMT.

Another simplification is inherent to the TectoMT's design: the English analysis code is shared for all language pairs with the English source side and similarly, the English synthesis code is shared for all systems translating into English. Only minor adaptations were needed here. For example, originally the English synthesis contained a block for insertion of definite and indefinite articles (*the, a*) based on heuristic rules. We replaced this synthesis block by a much simpler block that assigns articles based on the definiteness grammateme.[34] For source languages that have an explicit notion of definiteness (i.e. use indefinite and definite articles, either as separate words or suffixes), it is much more straightforward to use the same definiteness value in English and change it only when needed. Czech is the only exception among the five languages, as no articles are used and definiteness is only implicit. For Czech→English translation, the definiteness grammateme must be thus assigned in the transfer phase.

[32] `http://qtleap.eu`

[33] Parts of this section were published in Dušek et al. [2015b]. More details about the systems can be found in Popel et al. [2015].

[34] We have added the definiteness grammateme on the t-layer because it is easier to work with in translation than encoding the definiteness in deep word order and contextual boundness/non-boundness from the FGD theory.

The t-layer representation mostly follows the Czech and English TectoMT annotation style, which is in turn based on PDT and PCEDT [Hajič et al., 2012] treebanks. We adapted t-layer guidelines for some language-specific phenomena. For example, in Spanish block `A2T::ES::SetFormeme`, we introduced a new formeme `adj:left-attr`, which encodes the *adjective-noun* word order, which is less frequent in Spanish and may indicate syntactic or semantic distinction relative to the standard *noun-adjective* word order (where the adjective is assigned the standard formeme `adj:attr`). We are aware of several insufficiencies of the current t-layer style for some languages, e.g. the current set of grammatemes cannot express all required morphological meanings.

For the debugging and testing of the new analysis and synthesis pipelines we used monolingual "roundtrip" experiments: we analyzed the development data up to the t-layer and then synthesized back to word forms. We compared the resulting sentences with the original ones and focused on the differences. This allowed us to quickly reveal errors that could otherwise remain unnoticed for a long time (and deteriorate the translation). Also, this experiment can be done before the translation models are trained. Moreover, by improving the analysis phase, we actually improve also the translation models' quality.

All the newly added language pairs in TectoMT were tuned for translating IT-domain texts, so their BLEU evaluation is provided in the following section on domain adaptation. According to Tables 3.7–3.6, TectoMT outperformed Moses baseline for 5 out of the 10 language pairs: cs→en, nl→en, en→cs, en→es and en→pt.

3.7 Domain Adaptation

The real-use scenario of the QTLeap project was a multilingual helpdesk interface for IT troubleshooting: A user asks a question over a chat in Czech, Spanish, Portuguese, Dutch or Basque. The question is automatically translated to English using TectoMT. An information retrieval system finds the best matching answer in a database of questions and answers (or alerts a human operator if no suitable answer is found). The answer is then translated from English back to the language of the given user.

3.7.1 Baselines

For each language pair, we had relatively large general-domain parallel training data, but only a limited amount of in-domain data [see Burchardt and Avramidis, 2015, for details]. In the experiments reported in this section, the in-domain training data consisted of only 1000 questions (Batch1q) and 1000 answers (Batch1a).

A baseline SMT system (Moses) for each language pair was trained using the general-domain data and tuning hyper-parameters (with MERT [Och, 2003]) on Batch1, which is a standard form of baseline domain adaptation for SMT.[35] TectoMT baseline systems were prepared as described in Section 3.6, training the translation models on the general-domain data only and using Batch1 as a development set for tracking progress, but not for extensive tuning.

3.7.2 Domain-Adaptation Techniques Used

Improved translation of questions and imperatives

Our general-domain parallel training and development data did not contain many questions and imperatives. Also the taggers and parsers in TectoMT were trained on sentences with a limited amount of questions and imperatives. As a result, our initial TectoMT translations contained many errors related to questions and imperatives. For example, words such as *use*, *check*, *hit* etc. were tagged (and translated) as nouns although used as imperative verbs. We implemented rule-based blocks focused on fixing these errors as well as generating t-nodes for dropped subjects (with 2nd person) of English imperatives and fixing the word order of questions in the English synthesis phase. Most of these blocks were included already in the version marked as “TectoMT baseline” in Table 3.7, so we do not report any evaluation of these improvements.

HideIT – not translating specific expressions

Specific expressions from the IT domain such as URLs, email addresses, Unix commands and paths should not be translated. We implemented a heuristic

[35] In Rosa et al. [2016], we evaluated also other forms of domain adaptation for SMT, e.g. extracting a secondary phrase table from the in-domain data.

detection of these expressions and a simple mechanism which replaces these expressions with placeholders. After the translation the placeholders are replaced with the original expressions. We call this approach HideIT; see Agirre et al. [2016] for details.

Gazetteer – translating specific expressions

Other kind of expressions such as menu items, button names, messages and product names (e.g. *Class All Tabs*, *Cancel*, *Press enter to continue*, *MS Word*) need to be translated (and sometimes even *localized*, i.e. adapted to the target locale/country). We used a bilingual database (gazetteer) of IT domain expressions, provided within the WMT IT-domain translation task[36] [Bojar et al., 2016b]. We call this approach Gazetter; see Rosa et al. [2016] for details.

MaxEnt TM interpolation

In addition to the MaxEnt TM (described in Section 3.3) trained on general-domain data, we trained another MaxEnt TM on the in-domain training data. We interpolated the translation probabilities of these two models. See Rosa et al. [2015] for details.

VowpalWabbit TM fine-tuning

The VowpalWabbit TM described in Section 3.4 was trained with two passes of online learning on the general-domain data (CzEng). We took this saved model and continued training it with two more passes on the in-domain data (Batch1a). The number of passes was tuned in a previous cross-validation experiment. Batch1a is much smaller than CzEng (one thousand sentences versus 15 million sentences), but online training is more sensitive to the later training examples, so this approach is reasonably effective.

[36] `http://www.statmt.org/wmt16/it-translation-task.html`

3.7.3 Evaluation

Tables 3.6 and 3.7 present BLEU results of our domain-adaptation experiments for translation from English and to English, respectively. The "TectoMT baseline" row shows BLEU scores of TectoMT where all the domain-adaptation components (except for the improvements of questions and imperatives) are switched off. The rows below present the effect of switching on each of the domain-adaptation components relative to the TectoMT baseline. The Vowpal-Wabbit TM fine-tuning was performed only for en→cs. For en→pt, en→eu and en↔es, also other techniques were applied [see Agirre et al., 2016].

The row "Δ total" shows the effect of switching on all the components. Note that this is not a sum of the deltas for individual components because the effects of these components may overlap. Some of these overlaps are systematic: by activating the VowpalWabbit TM fine-tuning, we deactivate the MaxEnt TM interpolation. Thus, "Δ MaxEnt TM interpolation" cannot be combined with "Δ VowpalWabbit TM fine-tuning". We can see that the latter is more effective (+2.35 BLEU) than the former (+0.78 BLEU). In other words, the VowpalWabbit-based domain adaptation is 1.57 BLEU better than the MaxEnt-based domain adaptation.[37]

	BLEU (QTLeap Batch2a)				
system	en→cs	en→es	en→eu	en→nl	en→pt
Moses	31.07	25.11	**28.37**	**32.94**	19.36
TectoMT baseline	28.84	22.41	16.07	25.01	20.33
Δ HideIT	+0.79	+0.49	+1.00	+0.74	+0.37
Δ Gazetteer	+3.67	+3.47	+2.98	+3.02	+1.00
Δ MaxEnt TM interpolation	+0.78	+6.75	+0.17	+0.92	+1.42
Δ VowpalWabbit TM fine-tuning	+2.35				
Δ other		+3.84	+2.50		+0.44
Δ total	+5.83	+11.77	+7.34	+4.60	+3.12
TectoMT final	**34.67**	**34.18**	23.41	29.61	**23.45**

Table 3.6: Translations from English (Batch2a). Effect of various domain-adaptation modules on BLEU performance.

[37] Our MaxEnt implementation is trained with *batch learning*, so the online-learning fine-tuning is not applicable.

	BLEU (QTLeap Batch2q)				
system	cs→en	es→en	eu→en	nl→en	pt→en
Moses	26.44	**39.30**	**25.29**	36.45	**22.59**
TectoMT baseline	29.02	24.85	14.12	44.46	12.77
Δ HideIT	+0.00	+0.00	+0.06	+0.00	+0.02
Δ Gazetteer	+0.89	+0.59	+0.00	−0.34	+0.03
Δ MaxEnt TM interpolation	+1.14	+1.09	+1.48	+1.85	+1.66
Δ other		+2.03			
Δ total	+2.12	+3.15	+1.44	+1.85	+1.71
TectoMT final	**31.14**	28.00	15.56	**46.31**	14.48

Table 3.7: Translations to English (Batch2q). Effect of various domain-adaptation modules on BLEU performance. The best result for each language pair is marked in bold.

3.8 Advantages and Disadvantages of TectoMT

Advantages

- We think the main advantage of TectoMT over SMT is an explicit modeling of the hierarchical sentence structure. TectoMT uses explicit linguistic structures (dependency trees) on both the source and target side. TectoMT translations are very different from SMT translations. The complementary qualities of TectoMT and SMT (cf. Section 2.3) allowed to build the Chimera combination, which was the best English-to-Czech MT system in WMT 2013–2015.

- Tectogrammatical trees for the same sentence are more similar across languages relative to surface-dependency trees (a-trees in terms of PDT) [Mareček et al., 2008]. Consequently, the transfer is easier. The analysis and synthesis phases can be reused for many translation directions (e.g. the English analysis in TectoMT was implemented originally for en→cs, but later reused for en→es, en→eu, en→nland en→pt). This is the underlying motivation of all transfer-based MT approaches [Vauquois, 1975].

- Formemes seem to be an important feature of TectoMT: On the one hand, they support the deep-syntactic transfer by being mapped one-to-one to t-nodes and representing some kind of generalization compared to a plain list of function words. On the other hand, they are still surface-oriented, i.e. the mapping between surface text and formemes is relatively reliably predictable in both direction (analysis and synthesis).

- TectoMT allows to divide the whole translation process into meaningful parts: the three main phases analysis–transfer–synthesis are further divided into subtasks up to the level of Treex *blocks*. These are usually linguistically interpretable and can be developed relatively independently. This also allows to track down each translation error to a particular block in the pipeline.

- It is relatively easy to adapt TectoMT for a given application, without a need to retrain all the parts (cf. imperatives and questions in Section 3.7).

Disadvantages

Ideally, we would like to distinguish disadvantages inherently present in the design of TectoMT (which are very difficult or impossible to overcome) and disadvantages of the current implementation (which could be improved if more effort had been invested). However, both are closely interconnected.

- Explicit syntax modeling forces us to handle phenomena which are not necessary for the MT purposes. Often these phenomena are annotated in very different ways. For example, some PP-attachment ambiguities may be present in both source and target sentence, but in the dependency parsing we must choose only one representation, which is then used for the translation.[38] This makes our training data more sparse and we lose the alternative representation's context, which could help to disambiguate the translation. Another example is the strict boundary between functional and content words – we are forced to decide whether e.g. *by means of*, *by way of* or *with the help of* should be treated as multiword prepositions and encoded in the formeme value, or whether *means*, *way* and *help* should be treated as content words and annotated as a separate t-node.[39] Any boundary will cause "inconsistencies" between the source and target language, which result in the necessity of non-isomorphic transfer.[40]

- TectoMT uses 1-best analysis in tagging, parsing, transfer and synthesis.[41] SMT systems use beam search and consider thousands of possible translations, so e.g. translation ambiguities may be resolved using the language model. It would be very difficult to allow multiple dependency analyses to be stored in a compact form (similarly to forests or lattices in constituency parsing), while keeping the intuitive API for rule-based blocks.
- TectoMT does not use n-gram (nor neural) language model. Instead, it uses a target-language tree model (TreeLM). While n-gram LM is trained on the "correct" data, TreeLM is trained on automatically analyzed data and the automatic analysis introduces a bias (especially if the analysis of source and target language are incompatible in some aspects).
- While the modularity of TectoMT is an advantage, it comes also with a drawback: The whole system is very complex and the complexity tends to grow over the years. Fixing an error in one block can sometimes result in worsening of the translation because of an unexpected interference with a seemingly unrelated block.
- It is difficult to find enough skilled developers of TectoMT because the skills include (Perl) programming, machine learning and linguistics with at least a basic overview of tectogrammatics. This is a much rarer combination than (Python) programming and neural networks within TensorFlow.

[38] PP stands for *prepositional phrase*. For example, in "*Bring the box from the cellar!*" it is syntactically unclear whether the prepositional phrase *from the cellar* depends on the *box* or on the verb *bring*. Theoretically, there may be a *box from the cellar*, which is currently located in the garden, but if there is just one box in the cellar, then both syntactic structures have the same pragmatic meaning. In this case, the same (spurious) ambiguity is in the Czech translation ("*Přines tu bednu ze sklepa!*").

[39] In the current TectoMT implementation, only the first of the three phrases is treated as a multiword preposition.

[40] For example, in some contexts *with the help of*, *using* and *by* can be used interchangeably and translated to Czech either as *s pomocí* or *pomocí* or using the instrumental case without any preposition.

[41] Inside the lemma and formeme translation, multiple variants are considered and several re-ranking blocks are applied (including HMTM—Hidden Markov Tree Model), but after the transfer phase, only the best variant is kept and used as input for the synthesis.

Chapter 4

Training NMT

Add one very large bilingual corpus,
one sentence-to-sentence alignment process,
a gallon of sophisticated Bayesian statistics,
a touch of pixie dust, and throw them
into the computational cauldron.
Lo and behold, out comes a self-generated, robust,
corpus-based, general-purpose machine translation system.
Who needs a dictionary, grammars, semantics, or linguists?...
This, of course, is a caricature of the statistical MT position.
However, remove the pixie dust, and the caricature
comes uncomfortably close to reality.
...
Neural nets have been suggested as an answer
to many of the World's problems,
so why not use them for MT as well?
Carbonell et al. [1992]

This chapter describes our experiments with NMT using the Tensor2Tensor framework[1] (abbreviated T2T) and the Transformer sequence-to-sequence model [Vaswani et al., 2017].[2] We examine some of the critical parameters that affect the final translation quality, memory usage, training stability and training time. The know-how presented in this chapter together with the novel training strategies presented in the following chapter (5) allowed us to build English→Czech and Czech→English NMT systems that outperformed all other systems and even human translation in WMT2018, as described in Chapter 6.

While investigations into the effect of hyper-parameters like learning rate and batch size are available in the deep-learning community [e.g. Bottou et al., 2016, Smith and Le, 2017, Jastrzebski et al., 2017], these are either mostly theoretic or experimentally supported from domains like image recognition rather than MT. The Transformer model is rather different from the previously used NMT models (based on RNN or convolutions, cf. Section 2.5), so it is not clear whether the training strategies developed for these models will work optimally also for the

[1] `https://github.com/tensorflow/tensor2tensor`

[2] This chapter is an adapted version of Popel and Bojar [2018]. For a brief introduction into NMT and a description of the Transformer model see Section 2.5.

Transformer model. Moreover, both the Transformer model and the T2T toolkit have been released only recently – in June 2017, so there is a very small number of relevant publications and experimental results as of June 2018. We are aware only of the following publications: Vaswani et al. [2017] explore the effect of number of layers, number of attention heads, dropout and dimensions of various parameters. Shaw et al. [2018] substitute the absolute position encoding with a relative-position-aware self-attention. Roy et al. [2018] describe techniques for faster inference. Shazeer and Stern [2018] introduce Adafactor optimization into Transformer.

In this chapter, we fill this gap by focusing exclusively on MT and on the Transformer model, providing potential best practices for this particular setting. We also focus on the English-Czech language pair with much larger training data (58 million sentence pairs) than the previously reported experiments (English-German with 4.5 million sentence pairs and English-French 36 million sentence pairs [Vaswani et al., 2017]).

Some of our observations confirm the previous experience in the ML field, such as that larger training data are generally beneficial. Some of our observations are somewhat surprising, e.g. that training a model until achieving a given translation quality (as measured by BLEU) is more than three times faster on two GPUs relative to a single GPU (Section 4.3.7). Similarly interesting are our findings about the interaction between maximum sentence length, learning rate and batch size.

The rest of this chapter is structured as follows. In Section 4.1, we discuss our evaluation methodology and main criteria: translation quality and speed of training. Section 4.2 describes our dataset and its preparations. Section 4.3 is the main contribution of the chapter: a set of commented experiments, each wrapped with a set of conclusions and recommendations. Finally, Section 4.4 compares our best Transformer run with systems participating in WMT2017. We conclude in Section 4.5.

4.1 Evaluation Methodology

For automatic evaluation, we use the BLEU metric [Papineni et al., 2002], see Section 2.2.2 for details and a discussion on drawbacks of BLEU. We evaluated all our results in this chapter also with a character-based metric chrF [Popović, 2015], but the results were highly correlated with BLEU, so we omit these scores.

We implemented two helper script, `t2t-bleu` and `t2t-translate-all`, which we use for BLEU evaluation and plotting of all the learning curves presented in this chapter.

4.1.1 Considerations on Stopping Criterion

Training of NMT systems is usually non-deterministic,[3] and (esp. with the most recent models) hardly ever converges or starts overfitting[4] on reasonably large datasets. This leads to learning curves that never fully flatten, let alone start decreasing (see Section 4.3.2). The common practice of machine learning to evaluate the model on a final test set when it started overfitting (or shortly before that moment) is thus not applicable in practice.

Many papers in NMT do not specify any stopping criteria whatsoever. In other cases, they mention only an approximate number of days the model was trained for, e.g. Bahdanau et al. [2014], sometimes the exact number of training steps is given but no indication on "how much converged" the model was at that point, e.g. Vaswani et al. [2017]. Most probably, the training was run until no further improvements were "clearly apparent" on the development test set, and the model was evaluated at that point. Such an approximate stopping criterion is rather risky: it is conceivable that different setups were stopped at different stages of training and their comparison is not fair.

A somewhat more reliable method is to keep training for a specified number of iterations or a certain number of epochs. This is however not a perfect solution either, if the training is stopped before reaching the maximal BLEU. It is quite possible that e.g. a more complex model would need a few more epochs and eventually arrived at a higher score than its competitor. Also, the duration of one training step (or one epoch) differs between models (see Section 4.3.1) and from the practical point of view, we are mostly interested in the wall-clock time.

We tried the standard technique of *early stopping*, which means that the training is stopped when N subsequent evaluations on the development test set do not give improvements larger than a given delta. We saw a large variance in the training time and final BLEU, when applying early stopping, even for experiments with the same hyper-parameters and only a different random seed. Moreover to get the best results, we would have had to use a very large N and a very small delta.

[3] Even if we fix the random seed, a change of some hyper-parameters may affect the results not because of the change itself, but because it influenced the random initialization.

[4] By overfitting we mean here that the translation quality (test-set BLEU) begins to worsen, while the training loss keeps improving.

Based on the discussion above, we decided to report always the full learning curves and not only single scores. This solution does not fully prevent the risk of premature judgments, but the readers can at least judge for themselves if they would expect any sudden twist in the results or not.

In all cases, we plot the case-insensitive BLEU score (see Section 2.2.2) against the wall-clock time in hours. This solution obviously depends on the hardware chosen, so we always used the same equipment: one up to eight GeForce GTX 1080 Ti GPUs with NVIDIA driver 375.66 on Intel Xeon E5-2620. Some variation in the measurements is unfortunately unavoidable because we could not fully isolate the computation from different processes on the same machine and from general network traffic, but based on our experiments with replicated experiments such variation is negligible.

4.1.2 Terminology

For clarity, we define the following terms and adhere to them for the rest of the chapter (see also Section 2.5 for a general introduction to NMT):

Translation quality is an automatic estimate of how well the translation carried out by a particular fixed model expresses the meaning of the source text. We estimate translation quality by BLEU score (cf. Section 2.2.2).

Training Steps denote the number of iterations, i.e. the number of times the optimizer update was run. This number is also equal to the number of (mini)batches that were processed.

Batch Size is the number of training examples used by one GPU in one training step. In sequence-to-sequence models, batch size is usually specified as the number of *sentence pairs*. However, the parameter `batch_size` in T2T translation specifies the approximate number of *tokens* (subwords) in one batch.[5] This allows to use a higher number of short sentences in one batch or a smaller number of long sentences.

Effective Batch Size is the number of training examples consumed in one training step. When training on multiple GPUs, the parameter `batch_size` is interpreted per GPU, i.e. with `batch_size=1500` and 8 GPUs, the system actually digests 12 thousand subwords of each language in one step.

Training Epoch corresponds to one complete pass over the training data. Unfortunately, it is not easy to measure the number of training epochs in T2T.[6] T2T reports only the number of training steps. In order to convert training steps to epochs, we need to multiply the steps by the effective

[5] For this purpose, the number of tokens in a sentence is defined as the maximum of source and target subwords. T2T also does reordering and bucketing of the sentences by their length to minimize the use of padding symbols. Some padding is still needed, thus `batch_size` only approximates the actual number of (non-padding) subwords per batch.

[6] https://github.com/tensorflow/tensor2tensor/issues/415

batch size and divide by the number of subwords in the training data (see Section 4.2.1). The segmentation of the training data into subwords is usually hidden to the user and the number of subwords must be thus computed by a special script.

Computation Speed is simply the observed number of training steps per hour. Computation speed obviously depends on the hardware (GPU speed, GPU-CPU communication) and software (driver version, CUDA library version, implementation). The main parameters affecting computation speed are the model size, optimizer and other settings that directly modify the formula of the neural network.

Training Throughput is the amount of training data digested by the training. We report training throughput in subwords per hour. Training Throughput equals to the Computation Speed multiplied by the effective batch size.

Convergence Speed or **BLEU Convergence** is the increase in BLEU in a given time span (e.g. one hour or one day). Convergence speed changes heavily during training, starting very high and decreasing as the training progresses. A converged model should have convergence speed of zero.

Time Until Score is the training time needed to achieve a certain level of translation quality, in our case BLEU. We use this as an informal measure because it is not clear how to define the moment of "achieving" a given BLEU score. We define it as time after which the BLEU never falls below the given level.[7]

Examples Until Score is the number of training examples (in subwords) needed to achieve a certain level of BLEU. It equals to the Time Until Score multiplied by Training Throughput.

4.2 Data Selection and Preprocessing

Our training data comes from CzEng 1.7 and three smaller sources (Europarl, News Commentary, Common Crawl), see Table 2.1 on page 17 for details. We use this dataset of 58M sentence pairs for most our experiments in this chapter. In some experiments (in Sections 4.3.2 and 4.3.6), we substitute CzEng 1.7 with an older and considerably smaller CzEng 1.0 [Bojar et al., 2012] containing 15M sentence pairs (so together with the three smaller sources, the total size is 16M).

To plot the performance throughout the training, we use `wmt13` as a development set. In Section 4.4, we apply our best model (judged from the performance on the development set) to the `wmt17` test set, for comparison with the state-of-the-art systems.

[7] Such definition of Time Until Score leads to a high variance of its values because of the relatively high BLEU variance between subsequent checkpoints (visible as a "flickering" of the learning curves in the figures). To decrease the variation one can use a larger development test set.

4.2.1 Training Data Preprocessing

T2T supports two types of tokenization into subword units (see Section 2.5.1). The tokenization is either provided by an external script (e.g. the popular BPE algorithm [Sennrich et al., 2016b]) or a built-in tokenization is used. The T2T built-in subword tokenization is based on the word-piece algorithm [Wu et al., 2016]. Unlike BPE it does not expect the input to be pre-tokenized to words and it can fully recover the original raw text including spaces.[8] Based on a small sample of the training data, T2T will train a subword vocabulary and apply it to all the training and later evaluation data.

We use the T2T built-in tokenization and provide raw plain-text training sentences. We use the default parameters: shared source and target (English and Czech) subword vocabulary of size 32k. After this preprocessing, the total number of subwords in our main training data is 992 millions (taking the maximum of English and Czech lengths for each sentence pair, as needed for computing the number of epochs; see Section 4.1.2). The smaller dataset CzEng 1.0 has 327 million subwords. In both cases the average number of subwords per (space-delimited) word is ca. 1.5.

Even when following the default parameters, there are important details to be considered:

- We make sure that the subword vocabulary is trained on a sufficiently large sample of the training data.[9]
- As discussed in Section 4.3.5, a larger batch size may be beneficial for the training and the batch size can be larger when excluding training sentences longer than a given threshold. This can be controlled with parameter `max_length` (see Section 4.3.4), but it may be a good idea to exclude too long sentences even before preparing the training data using `t2t-datagen`. This way the TFRecords training files will be smaller and their processing a bit faster.[10]

4.3 Experiments

In this section, we present several experiments, always summarizing the observations and conclusions from each experiment. All experiments in this chapter were done with T2T v1.2.9 unless stated otherwise.

We experiment with two sets of hyper-parameters which are pre-defined in T2T: `transformer_big_single_gpu` (BIG) and `transformer_base_single_gpu`

[8] There is also a difference in encoding whether a given subword is followed by a space. For details, see Macháček et al. [2018] who report a 4.9 BLEU improvement on German→Czech translation caused by switching from the default BPE subwords to T2T built-in subwords.

[9] This is controlled by a `file_byte_budget` constant, which must be changed directly in the source code in T2T v1.2.9. A sign of too small training data for the subword vocabulary is that the `min_count` as reported in the logs is too low, so the vocabulary is estimated from words seen only once or twice.

[10] We did no such pre-filtering in our experiments in this chapter (cf. Section 5.3.10).

(BASE). They differ mainly in the size of the model. Note that `transformer_big_single_gpu` and `transformer_base_single_gpu` are only names of a set of hyper-parameters, which can be applied even when training on multiple GPUs, as we do in our experiments; see Section 4.3.7.[11]

Our baseline setting uses the BIG model with its default hyper-parameters except for:

- `batch_size=1500` (see the discussion of different sizes in Section 4.3.5),
- `--train_steps=6000000`, i.e. high enough, so we can stop each experiment manually as needed,
- `--save_checkpoints_secs=3600` which forces checkpoint saving each hour (see Section 4.3.10),
- `--schedule=train` which disables the internal evaluation with `approx_bleu` and thus makes training a bit faster.[12]

4.3.1 Computation Speed and Training Throughput

We are primarily interested in the translation quality (BLEU learning curves and Time Until Score) and we discuss it in the following sections 4.3.2–4.3.10. In this section, we focus however only on the *computation speed* and *training throughput*. Both are affected by three important factors: batch size, number of used GPUs and model size. The speed is usually almost constant for a given experiment.[13]

Table 4.1 shows the computation speed and training throughput for a single GPU and various batch sizes and model sizes (BASE and BIG). The BASE model allows for using a higher batch size than the BIG model. The cells where the BIG model resulted in out-of-memory errors are marked with "OOM".[14] We can see that the computation speed decreases with increasing batch size because not all operations in GPU are fully batch-parallelizable. The training throughput grows sub-linearly with increasing batch size, so based on these experiments only, there is only a small advantage when setting the batch size to the maximum value. We will return to this question in Section 4.3.5, while taking into account the translation quality.

[11] According to our experiments (not reported here), `transformer_big_single_gpu` is better than `transformer_big` even when training on 8 GPUs, although the naming suggests that the T2T authors had an opposite experience.

[12] Also there are some problems with the alternative schedules `train_and_evaluate` (it needs more memory) and `continuous_train_and_eval` (see `https://github.com/tensorflow/tensor2tensor/issues/556`).

[13] TensorBoard shows `global_step/sec` statistics, i.e. the computation speed curve. These curves in our experiments are almost constant for the whole training with variation within 2%, except for moments when a checkpoint is being saved (and the computation speed is thus much slower).

[14] For these experiments, we used `max_length=50` in order to be able to test larger batch sizes. However, in additional experiments we checked that `max_length` does not affect the training throughput itself.

	model	
batch_size	BASE	BIG
500	43.4k	23.6k
1000	30.2k	13.5k
1500	22.3k	9.8k
2000	16.8k	7.5k
2500	14.4k	6.5k
3000	12.3k	OOM
4500	8.2k	OOM
6000	6.6k	OOM

(a) Computation speed (steps/hour)

	model	
batch_size	BASE	BIG
500	21.7M	11.9M
1000	30.2M	13.5M
1500	33.4M	14.7M
2000	33.7M	15.0M
2500	36.0M	16.2M
3000	37.0M	OOM
4500	36.7M	OOM
6000	39.4M	OOM

(b) Training throughput (subwords/hour)

Table 4.1: Computation speed and training throughput for a single GPU.

We can also see the BASE model has approximately two times larger throughput, as well as computation speed, relative to the BIG model.

GPUs	steps/hour	subwords/hour
1	9.8k	14.7M
2	7.4k	22.2M
6	5.4k	48.6M
8	5.6k	67.2M

Table 4.2: Computation speed and training throughput for various numbers of GPUs, with the BIG model and `batch_size=1500`. Note that (training) *steps* are defined in Section 4.1.2 as 'the number of times the optimizer update was run' and that the optimizer does synchronous updates in case of multiple GPUs.

Table 4.2 uses the BIG model and `batch_size=1500`, while varying the number of GPUs. The overhead in GPU synchronization is apparent from the decreasing computation speed. Nevertheless, the training throughput still grows with more GPUs, so e.g. with 6 GPUs we process 3.2 times more training data per hour relative to a single GPU (while without any overhead we would hypothetically expect 6 times more data).

The overhead when scaling to multiple GPUs is smaller than the overhead when scaling to a higher batch size. Scaling from a single GPU to 6 GPUs increases the throughput 3.2 times, but scaling from batch size 1000 to 6000 on a single GPU increases the throughput 1.3 times.

4.3.2 Training Data Size

For this experiment, we substituted CzEng 1.7 with CzEng 1.0 in the training data, so the total training size is 16 million sentence pairs (cf. Section 2.1).

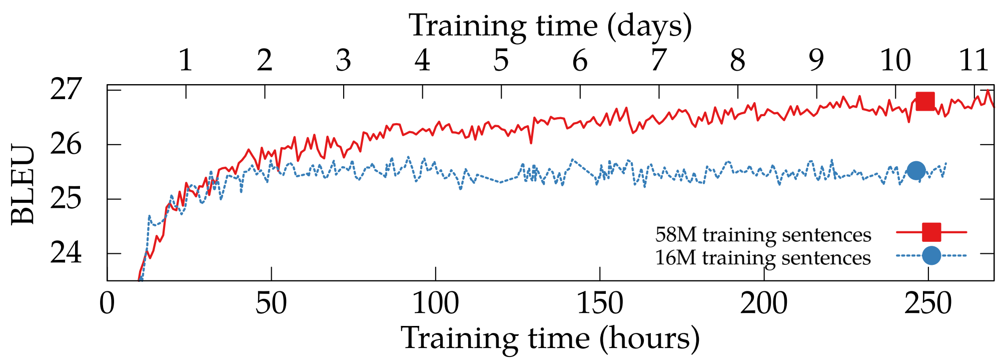


Figure 4.1: Training data size effect. BLEU learning curves for our main training dataset with 58 million sentence pairs and an alternative training dataset with 16 million sentence pairs. Both trained with 8 GPUs, BIG model and `batch_size=1500`.

Figure 4.1 compares the BLEU learning curves of two experiments which differ only in the training data: the baseline CzEng 1.7 versus the smaller CzEng 1.0. Both are trained on the same hardware with the same hyper-parameters (8 GPUs, BIG, `batch_size=1500`). Training on the smaller dataset (2.5 times smaller in the number of words) converges to BLEU of ca. 25.5 (on the `wmt13` dev set) after two days of training and does not improve over the next week of training. Training on the larger dataset gives slightly worse results in the first eight hours of training (not shown in the graph) but clearly better results after two days of training, reaching over 26.5 BLEU after eight days.[15]

With `batch_size=1500` and 8 GPUs, training one epoch of the smaller dataset (with CzEng 1.0) takes 27k steps (5 hours of training), compared to 83k steps (15 hours) for the larger dataset (with CzEng 1.7). This means ca. 10 epochs in the smaller dataset were needed for reaching the convergence and this is also the moment when the larger dataset starts being clearly better. However, even 18 epochs in the larger dataset were not enough to reach the convergence.

Conclusions Related to Training Data Size

- For comparing different datasets (e.g. smaller and cleaner vs. larger and noisier), we need to train the network for a sufficient amount of time, given that the results after first several hours (or days if training on a single GPU) are not representative of the ultimate performance.

- For large training data (e.g. CzEng 1.7, which has over 500M words), BLEU improves even after one week of training on eight GPUs (or after 20 days of training on two GPUs in another experiment).

[15] We compared the two datasets also in another experiment with two GPUs, where CzEng 1.7 gave slightly worse results than CzEng 1.0 during the first two days of training but clearly better results after eight days. We hypothesize CzEng 1.0 is somewhat cleaner than CzEng 1.7.

- We cannot simply linearly interpolate one dataset results to another dataset. While the smaller training data (with CzEng 1.0) converged after 2 days, the main training data (with CzEng 1.7), which is 2.5 times larger, continues improving even after 2.5×2 days.[16]

4.3.3 Model Size

Choosing the right model size is important for practical reasons: larger models may exceed the memory capacity of GPUs, or they may require to use a very small batch size.

We present experiments with two models,[17] which are pre-defined in Tensor2Tensor: `transformer_big_single_gpu` (BIG) and `transformer_base_single_gpu` (BASE). They differ in four hyper-parameters summarized in Table 4.3.

model	hidden_size	filter_size	num_heads	adam_beta2
BASE	512	2048	8	0.980
BIG	1024	4096	16	0.998

Table 4.3: Differences in hyper-parameter of `transformer_big_single_gpu` (BIG) and `transformer_base_single_gpu` (BASE). See Section 2.5.2 for explanation of the first three hyper-parameters (hidden_size corresponds to d_{model}, filter_size is the dimension of the feed-forward sublayer). `adam_beta2` is the second-moment hyper-parameter of the Adam optimizer [Kingma and Ba, 2014].

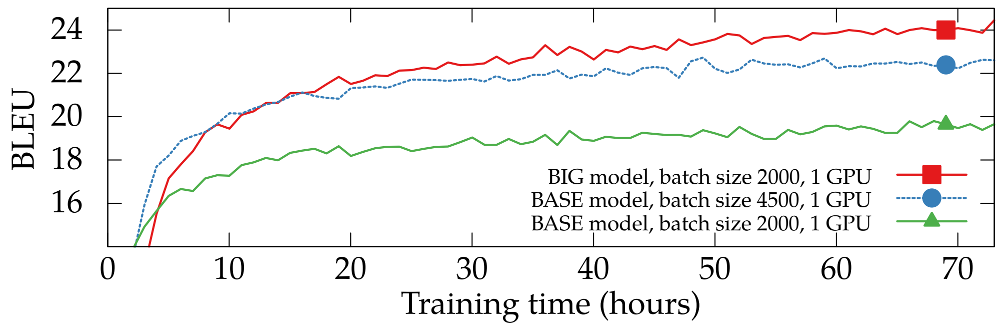


Figure 4.2: Effect of model size and batch size on a single GPU.

Figure 4.2 shows that on a single GPU, the BIG model becomes clearly better than the BASE model after 4 hours of training if we keep the batch size the same – 2000 (and we have confirmed it with 1500 in other experiments). However, the

[16] Although such an expectation may seem naïve, we can find it in literature. For example, Bottou [2012] in Section 4.2 writes: *"Expect the validation performance to plateau after a number of epochs roughly comparable to the number of epochs needed to reach this point on the small training set."*

[17] We tried also a model three times as large as BASE (1.5 times as large as BIG), but it did not reach better results than BIG, so we don't report it here.

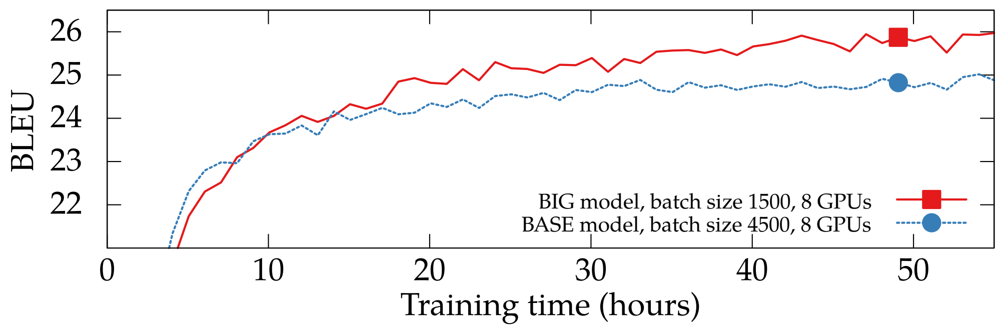


Figure 4.3: Effect of model size and batch size on 8 GPUs.

BASE model requires less memory, so we can afford a larger batch size, in our case 4500 (with no `max_length` restriction; see the next section), which improves the BLEU (see Section 4.3.5). But even so, after less than one day of training, BIG with batch size 2000 becomes better than BASE with batch size 4500 (or even 6000 with `max_length`=70 in another experiment) and the difference grows up to 1.8 BLEU after three days of training.

Figure 4.3 confirms this with 8 GPUs – here BIG with batch size 1500 becomes clearly better than BASE with batch size 4500 after 18 hours of training.

Conclusions Related to Model Size

- The BIG model is preferable over the BASE model if more than one day is used for training and 11GB (or more) GPU memory is available.
- With less memory, we recommend to benchmark BIG and BASE with the maximum possible batch size.

4.3.4 Maximum Training Sentence Length

The parameter `max_length` specifies the maximum sentence length in the number of subwords. Longer sentences (either in source or target language) are excluded from the training completely. If no `max_length` is specified (which is the default), `batch_size` is used instead. Lowering the `max_length` allows to use a larger batch size or a larger model. Since the Transformer implementation in T2T can suddenly run out of memory even after several hours of training, it is good to know how large batch size fits in our GPU. Table 4.4 presents what we empirically measured for the BASE and BIG models with Adam and Adafactor[18] optimizers and various `max_length` values.

	maximum batch size			longer sentences	
max_length	BIG+Adam	BIG+Adafactor	BASE+Adam	in train	in test
none	2040	2550	4950	0.0%	0.0%
150	2230	2970	5430	0.2%	0.0%
100	2390	3280	5990	0.7%	0.3%
70	2630	3590	6290	2.1%	2.2%
50	2750	3770	6430	5.0%	9.1%

Table 4.4: Maximum batch size which fits into 11GB memory for various combinations of `max_length` (maximum sentence length in subwords), model size (BASE or BIG) and optimizer (Adam or Adafactor). The last two columns show the percentage of sentences in the train (CzEng 1.7) and test (wmt13) data that are longer than a given threshold.

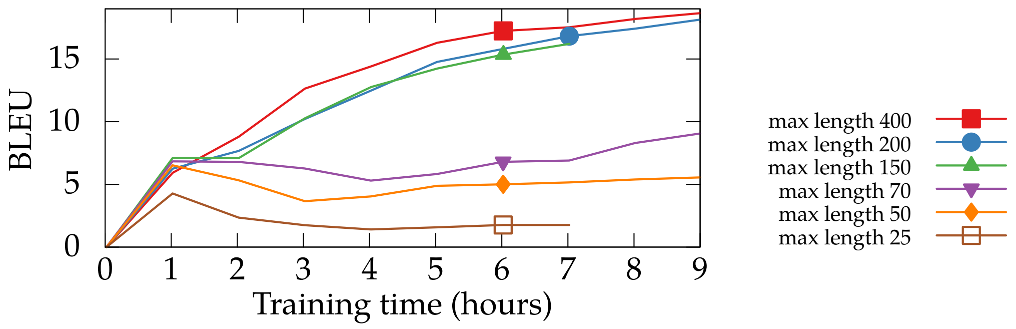


Figure 4.4: Effect of restricting the training data to various `max_length` values. All trained on 1 GPU with the BIG model and `batch_size`=1500. An experiment without any `max_length` is not shown, but it has the same curve as `max_length`=400.

Setting `max_length` too low would result in excluding too many training sentences and biasing the translation towards shorter sentences, which would hurt the translation quality. The last two columns in Table 4.4 show that setting

[18] The Adafactor optimizer [Shazeer and Stern, 2018] is available only in T2T 1.4.2 or newer and has three times smaller models than Adam because it does not store first and second moments for all weights. We use Adafactor in Chapter 5.

`max_length` to 70 results in excluding only 2.1% of sentences in the training data, and only 2.2% sentences in the development test data are longer. For `max_length` of 100, the number are 0.7% and 0.3%, respectively. Thus, the detrimental effect of smaller training data and length bias should be minimal in this setting. However, our experiments with `batch_size`=1500 in Figure 4.4 show an unexpected abrupt drop in BLEU after one hour of training for all experiments with `max_length` 70 or lower. Even with `max_length` 150 or 200, the BLEU learning curve is worse than with `max_length`=400, which finally gives the same result as not using any `max_length` restriction. The training loss of `max_length`=25 (and 50 and 70) has a high variance and stops improving after the first hour of training but shows no sudden increase (as in the case of diverged training discussed in Section 4.3.6 when the learning rate is too high). We have no explanation for this phenomenon.[19]

We performed another set of experiments with varying `max_length`, but this time with `batch_size`=2000 instead of 1500. In this case, `max_length` 25 and 50 still results in slower-growing BLEU curves, but for `max_length` of 70 and higher, the learning curve is essentially identical to the one of unrestricted `max_length`. Therefore, if the batch size is large enough, the `max_length` has almost no effect on BLEU, but this should be verified for each new dataset (obviously, all the results in this subsection are dependent on the distribution of sentence lengths in our training and test data).

We trained several models with various `max_length` values for three days and observed that they are not able to produce longer translations than what was the maximum length used in training, even if we change the decoding parameter alpha. Setting the alpha parameter too high results in longer translations, but only because word repetitions and nonsense translations are exploited. Our observation about the limited length of translations is in contrast with the hypothesis of Vaswani et al. [2017]: "*We chose the sinusoidal version* [of positional encoding] *because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.*"

Conclusions Related to `max_length`

- On the one hand, we recommend to set (a reasonably low) `max_length`. This allows to use a larger batch size and prevents out-of-memory errors after several hours of training. Also, with a higher percentage of training sentences that are almost `max_length` long, there is a higher chance that the training will fail either immediately (if the batch size is too large) or never (otherwise).[20]

- On the other hand, we recommend to set a reasonably high `max_length` by considering the percentage of sentences excluded from training and from the targeted development test set, and also by watching for unexpected drops (or stagnations) of the BLEU curve in the first hours of training.

[19] https://github.com/tensorflow/tensor2tensor/issues/582

[20] Note that by default, T2T shuffles all sentences in the training data.

4.3.5 Batch Size

The default `batch_size` value in recent T2T versions is 4096 subwords for all models except for `transformer_base_single_gpu`, where the default is 2048. However, we recommend to always set the batch size explicitly[21] or at least make a note what was the default in a given T2T version when reporting experimental results.

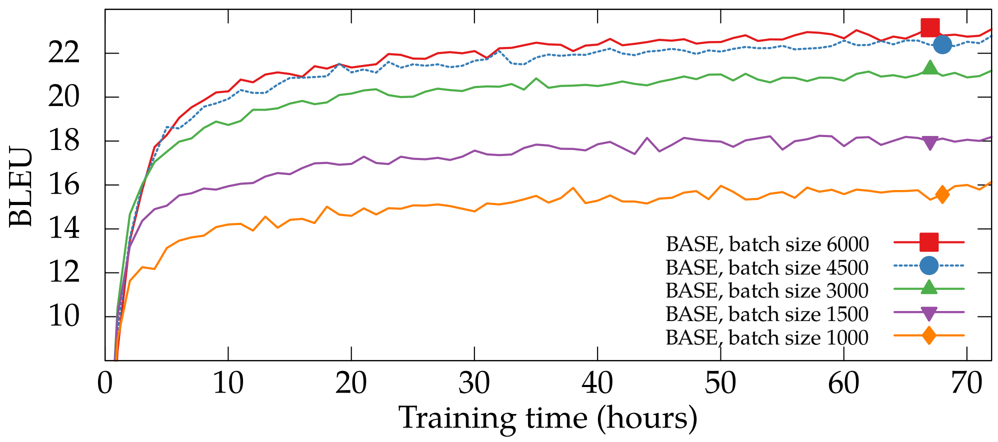


Figure 4.5: Effect of the batch size with the BASE model. All trained on 1 GPU.

Figure 4.5 shows learning curves for five different batch sizes (1000, 1500, 3000, 4500 and 6000) for experiments with a single GPU and the BASE model.[22] A larger batch size up to 4500 is clearly better in terms of BLEU as measured by Time Until Score and Examples Until Score metrics defined in Section 4.3.1. For example, to get over BLEU of 18 with `batch_size`=3000, we need 7 hours (260M examples), and with `batch_size`=1500, we need ca. 3 days (2260M examples), i.e. 10 times longer (9 time more examples). From Table 4.1a we know that larger batches have slower computation speed, so when re-plotting Figure 4.5 with steps instead of time on the x-axis, the difference between the curves would be even larger. From Table 4.1b we know that larger batches have slightly higher training throughput, so when re-plotting with the number of examples processed on the x-axis, the difference will be smaller, but still visible. The only exception is the difference between batch size 4500 and 6000 (in Figure 4.5), which is very small and can be fully explained by the fact that batch size 6000 has 7% higher throughput than batch size 4500.

In summary, a larger batch size yields better results when using the BASE model, although with diminishing returns. This observation goes against the common knowledge in other NMT frameworks and deep learning in general [e.g. Keskar et al., 2017] that smaller batches proceed slower (training examples

[21] e.g. `--hparams="batch_size=1500"`; as the batch size is specified in subwords, we see no advantage in using power-of-two values.

[22] All the experiments in Figure 4.5 use `max_length`=70, but we have obtained the same curves when re-running without any `max_length` restrictions, except for `batch_size`=6000 which failed with OOM.

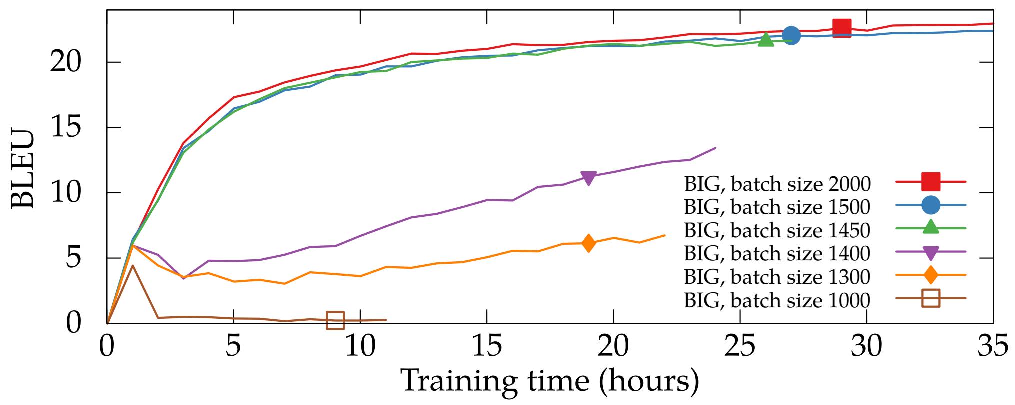


Figure 4.6: Effect of the batch size with the BIG model. All trained on a single GPU.

per hour) but result in better generalization (higher test-set BLEU) in the end. In our experiments with the BASE model in T2T, larger batches are not only faster in training throughput (as could be expected), but also faster in convergence speed, Time Until Score and Examples Until Score.

Interestingly, when replicating these experiments with the BIG model, we see quite different results, as shown in Figure 4.6. The BIG model needs a certain minimal batch size to start converging at all, but for larger batch sizes there is almost no difference in the BLEU curves (however, an increased batch size never makes the BLEU worse in our experiments). In our case, the cut-off is between batch size 1450, which trains well, and 1400, which drops off after two hours of training, recovering only slowly.

According to Smith and Le [2017] and Smith et al. [2017], the *gradient noise scale*, i.e. scale of random fluctuations in the SGD (or Adam etc.) dynamics, is proportional to learning rate divided by the batch size (cf. Section 4.3.8). Thus, when decreasing the batch size, we increase the noise scale and the training may *diverge*. This may be either permanent, as in the case of batch size 1000 in Figure 4.6, or temporary, as in the case of batch size 1300 and 1400, where the BLEU continues to grow after the temporary drop, but much more slowly than the non-diverged curves.

We are not sure what causes the difference between the BASE and BIG models with regards to the sensitivity to batch size. One hypothesis is that the BIG model is more difficult to initialize and thus more sensitive to divergence in the early training phase. Also, while for BASE, increasing the batch size was highly beneficial until 4500, for BIG this limit may be below 1450, i.e. below the minimal batch size needed for preventing diverged training.

Conclusions Related to Batch Size

- Batch size should be set as large as possible while keeping a reserve for not hitting the out-of-memory errors. It is advisable to establish the largest possible batch size before starting the main and long training.

4.3.6 Learning Rate and Warmup Steps on a Single GPU

The default learning rate in T2T translation models is 0.20. Figure 4.7 shows that varying the value within range 0.05–0.25 makes almost no difference. Setting the learning rate too low (0.01) results in notably slower convergence. Setting the learning rate too high (0.30, not shown in the figure) results in *diverged* training, which means in this case that the learning curve starts growing as usual, but at one moment drops down almost to zero and stays there forever.

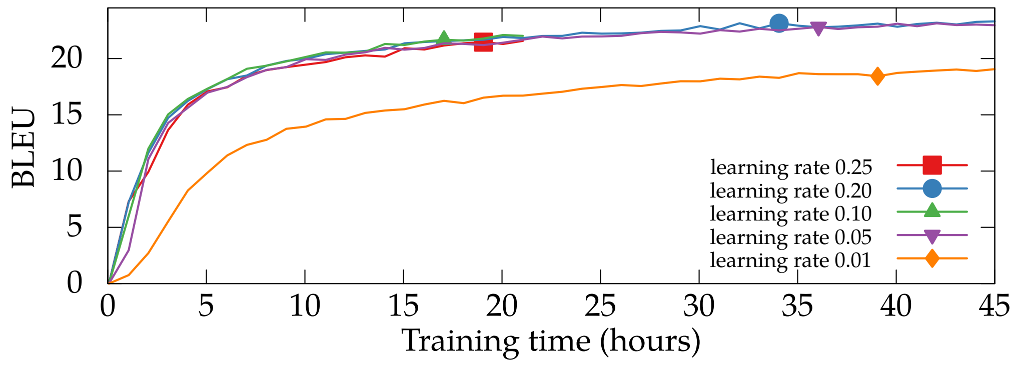


Figure 4.7: Effect of the learning rate on a single GPU. All trained on CzEng 1.0 with the default batch size (1500), BIG model and warmup steps (16k).

A common solution to prevent diverged training is to decrease the `learning_rate` parameter or increase `learning_rate_warmup_steps` or introduce gradient clipping. The `learning_rate_warmup_steps` parameter configures a `linear_warmup_rsqrt_decay` schedule[23] and it is set to 16 000 by default (for the BIG model), meaning that within the first 16k steps the learning rate grows linearly and then follows an inverse square root decay ($t^{-0.5}$, cf. Section 4.3.8). At 16k steps, the actual learning rate is thus the highest.

If a divergence is to happen, it usually happens within the first few hours of training, when the actual learning rate becomes the highest. Once we increased the warmup steps from 16k to 32k, we were able to train with the learning rate of 0.30 and even 0.50 without any divergence. The learning curves looked similar to the baseline one (with default values of 16k warmup steps and learning rate 0.20). When trying learning rate 1.0, we had to increase warmup steps to 60k (with 40k, the training diverged after one hour) – this resulted in a slower convergence at first (ca. 3 BLEU lower than the baseline after 8 hours of training), but after 3–4 days of training almost the same curve as in the baseline was achieved.

Figure 4.8 shows the effect of different warmup steps with a fixed learning rate (the default 0.20). Setting warmup steps too low (12k) results in diverged training. Setting them too high (48k, green curve) results in a slightly slower convergence at first, but matching the baseline after a few hours of training.

We can conclude that for a single GPU and the BIG model, there is a relatively large range of learning rate and warmup steps values that achieve the optimal

[23] The schedule was called `noam` in T2T versions older than 1.4.4. In recent T2T versions, it should be specified as `constant*linear_warmup*rsqrt_decay*rsqrt_hidden_size`.

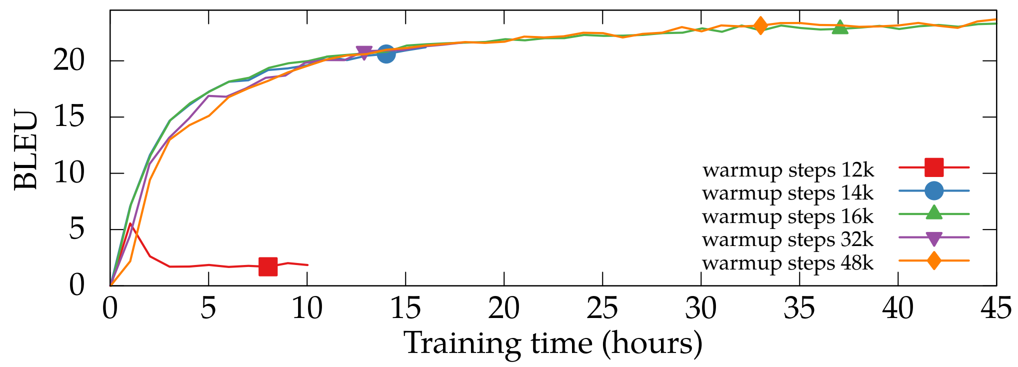


Figure 4.8: Effect of the warmup steps on a single GPU. All trained on CzEng 1.0 with the default batch size (1500) and learning rate (0.20).

results. The default values `learning_rate`=0.20 and `learning_rate_warmup_-steps`=16000 are within this range.

Conclusions Related to Learning Rate and Warmup Steps

- Gradient clipping and/or more warmup steps help to prevent diverged training.
- If that does not help (or if the warmup steps are too high relative to the expected total training steps), it is advisable to decrease the learning rate.
- Note that when decreasing the number of warmup steps (while maintaining the learning rate), the maximum actual learning rate is increased as well because of the way how the `linear_warmup_rsqrt_decay` schedule is implemented.

4.3.7 Number of GPUs

T2T allows to train with multiple GPUs on the same machine simply using the parameter `--worker_gpus`.[24] As explained in Section 4.1.2, the parameter `batch_size` is interpreted per GPU, so with 8 GPUs, the *effective batch size* is 8 times larger.

A single-GPU experiment with batch size 4000, should give exactly the same results as two GPUs and batch size 2000 and as four GPUs and batch size 1000 because the effective batch size is 4000 in all three cases. We have confirmed this empirically. By the "same results" we mean BLEU (or train loss) versus training steps on the x-axis. When considering time, the four-GPU experiment will be the fastest one, as explained in Section 4.3.1.

[24] and making sure environment variable `CUDA_VISIBLE_DEVICES` is set so enough GPUs are visible. T2T allows also distributed training (on multiple machines), but we have not experimented with it. Both single-machine multi-gpu and distributed training use synchronous Adam updates by default.

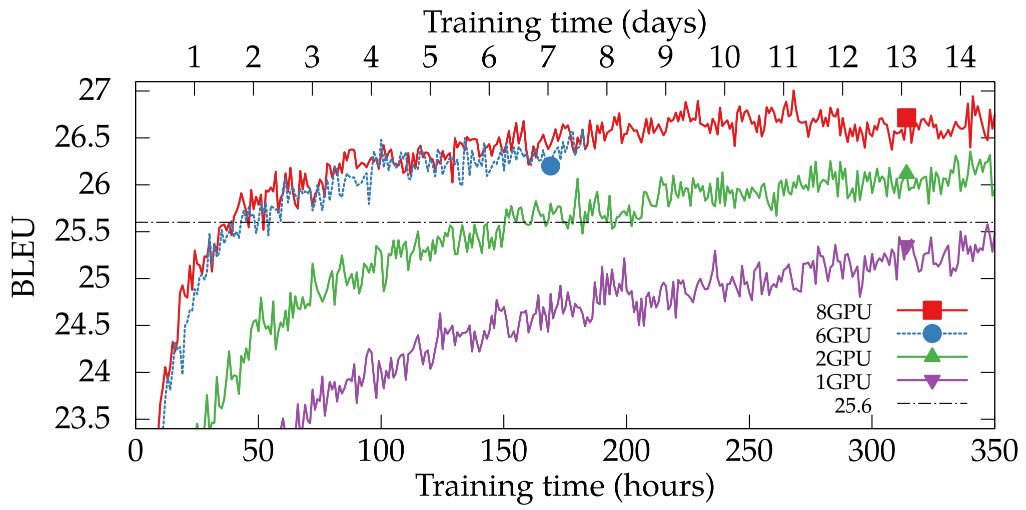


Figure 4.9: Effect of the number of GPUs. BLEU=25.6 is marked with a black line.

Figure 4.9 shows BLEU curves for different numbers of GPUs and the BIG model with batch size, learning rate and warmup steps fixed on their default values (1500, 0.20 and 16k, respectively). As expected, training with more GPUs converges faster. What is interesting is the Time Until Score. Table 4.5 lists the approximate training time and number of training examples (in millions of subwords) needed to "surpass" (i.e. achieve and never again fall below) BLEU of 25.6.

# GPUs	hours	subwords (M)
1	> 600	> 9000
2	203	$2322 \cdot 2 = 4644$
6	56	$451 \cdot 6 = 2706$
8	40	$341 \cdot 8 = 2728$

Table 4.5: Time and training data consumed to reach BLEU of 25.6, i.e. Time Until Score and Examples Until Score. Note that the experiment on 1 GPU was ended after 25 days of training without clearly surpassing the threshold (already outside of Figure 4.9).

We can see that two GPUs are more than three times faster than a single GPU when measuring the Time Until Score and need much less training examples (i.e. they have lower Examples Until Score). Similarly, eight GPUs are more than five times faster than two GPUs and 1.7 times less training data is needed.

Recall that in Figure 4.6 we have shown that increasing the batch size from 1450 to 2000 has almost no effect on the BLEU curve. However, when increasing the effective batch size by using more GPUs, the improvement is higher than could be expected from the higher throughput.[25] We find this quite surprising,

[25] It is possible to simulate multi-GPU training on a single GPU, simply by doing the update once after N batches (and summing the gradients). This is similar to the *ghost batches* of

especially considering the fact that we have not tuned the learning rate and warmup steps (see the next section).

Conclusions Related to the Number of GPUs

- For the fastest BLEU convergence, it is recommended to use as many GPUs as available (in our experiments up to 8).
- This holds even when there are more experiments to be done. For example, it is better to run one 8-GPUs experiment after another, rather than running two 4-GPUs experiments in parallel or eight single-GPU experiments in parallel.

4.3.8 Learning Rate and Warmup Steps on Multiple GPUs

Related Work

There is a growing number of articles on scaling deep learning to multiple machines with synchronous SGD (or its variants) by increasing the effective batch size. We will focus mostly on the question how to adapt the learning rate schedule, when scaling from one GPU (or any device, in general) to k GPUs.

Krizhevsky [2014] says "*Theory suggests that when multiplying the batch size by k, one should multiply the learning rate by $\sqrt{k}$ to keep the variance in the gradient expectation constant*", without actually explaining which theory suggests so. However, in the experimental part, he reports that what worked the best, was a *linear scaling heuristics*, i.e. multiplying the learning rate by k, again without any explanation nor details on the difference between $\sqrt{k}$ scaling and k scaling.

The linear scaling heuristics became popular, leading to good scaling results in practice [Goyal et al., 2017, Smith et al., 2017] and also theoretical explanations [Bottou et al., 2016, Smith and Le, 2017, Jastrzebski et al., 2017]. Smith and Le [2017] interpret SGD (and its variants) as a stochastic differential equation and show that the *gradient noise scale* g equals $\epsilon\left(\frac{N}{B}-1\right)$, where ϵ is the learning rate, N is the training set size, and B is the effective batch size. This noise "*drives SGD away from sharp minima, and therefore there is an optimal batch size which maximizes the test set accuracy*". In other words, for keeping the optimal level of gradient noise (which leads to "flat minima" that generalize well), we need to scale the learning rate linearly when increasing the effective batch size.

However, Hoffer et al. [2017] suggest to use $\sqrt{k}$ scaling instead of the linear scaling and provide both theoretical and empirical support for this claim. They show that $\mathrm{cov}(\Delta w, \Delta w) \propto \frac{\epsilon^2}{NB}$, thus if we want to keep the the covariance matrix of the parameters update step Δw in the same range for any effective batch size B, we need to scale the learning rate proportionally to the square root of

Hoffer et al. [2017], but using ghost batch size larger than the actual batch size. Independently on Popel and Bojar [2018], this was suggested by Saunders et al. [2018] under name *Delayed SGD update* and by Ott et al. [2018] under name *Accumulating gradients*. It is now available in T2T via the option `--optimizer_multistep_accumulate_steps` and it would be interesting to explore even larger effective batch sizes.

B. They found that $\sqrt{k}$ scaling works better than linear scaling on CIFAR10.[26] You et al. [2017] confirm linear scaling does not perform well on ImageNet and suggest to use Layer-wise Adaptive Rate Scaling.

We can see that finding the optimal design of large-batch training is still an open research question. Most of the articles cited above have experimental support only from the image recognition tasks (usually ImageNet) and convolutional networks (e.g. ResNet), so it is not clear whether their suggestions can be applied also to sequence-to-sequence tasks (NMT) with self-attentional networks (Transformer). There are several other differences as well: Modern convolutional networks are usually trained with *batch normalization* [Ioffe and Szegedy, 2015], which seems to be important for the scaling, while Transformer uses *layer normalization* [Lei Ba et al., 2016].[27] Also, Transformer uses Adam together with an inverse-square-root learning-rate decay, while most ImageNet papers use SGD with momentum and piecewise-constant learning-rate decay.

Our Experiments

We decided to find out empirically the optimal learning rate for training on 8 GPUs. Increasing the learning rate from 0.20 to 0.30 resulted in diverged training (BLEU dropped to almost 0 after two hours of training). Similarly to our single-GPU experiments (Section 4.3.6), we were able to prevent the divergence by increasing the warmup steps or by introducing gradient clipping (e.g. with `clip_grad_norm=1.0`, we prevented the divergence for learning rate 0.40, but increasing it further to 0.60 led to divergence nevertheless). However, none of these experiments led to any improvements over the default learning rate – all had about the same BLEU curve after few hours of training.

Jastrzebski et al. [2017] show that "*the invariance under simultaneous rescaling of learning rate and batch size breaks down if the learning rate gets too large or the batch size gets too small*". A similar observation was reported e.g. by Bottou et al. [2016]. Thus, our initial hypothesis was that 0.20 (or 0.25) is the maximal learning rate suitable for stable training in our experiments even when we scale from a single GPU to 8 GPUs. Considering this initial hypothesis, we were surprised that we were able to achieve such a good Time Until Score with 8 GPUs (more than 8 times smaller relative to a single GPU, as reported in Table 4.5). To answer this riddle, we need to understand how learning rate schedules are implemented in T2T.

[26] To close the gap between small-batch training and large-batch training, Hoffer et al. [2017] introduce (in addition to $\sqrt{k}$ scaling) so-called *ghost batch normalization* and *adapted training regime*, which means decaying the learning rate after a given number of steps instead of epochs.

[27] Applying batch normalization on RNN is difficult. Transformer does not use RNN, but still, we were not successful in switching to batch normalization (and possibly ghost batch normalization) due to NaN loss errors.

Parametrization of Learning Rate Schedules in T2T

In most works on learning rate schedules,[28] the "time" parameter is actually interpreted as the number of epochs or training examples. For example a popular setup for piecewise-constant decay in ImageNet training [e.g. Goyal et al., 2017] is to divide the learning rate by a factor of 10 at the 30-th, 60-th and 80-th epoch.

However, in T2T, it is the `global_step` variable that is used as the "time" parameter. So when increasing the effective batch size 8 times, e.g. by using 8 GPUs instead of a single GPU, the actual learning rate[29] achieves a given value after the same number of steps, but after 8 times fewer training examples. For the inverse-square-root decay, we have $\mathit{actual\_lr}(\mathit{steps}) = c \cdot \mathit{steps}^{-0.5} = \frac{1}{\sqrt{8}} \cdot \mathit{actual\_lr}(\mathit{steps} \cdot 8)$, where c is a constant containing also the `learning_rate` parameter. So with 8 GPUs, if we divide the `learning_rate` parameter by $\sqrt{8}$, we achieve the same actual learning rate after a given number of training examples as in the original single-GPU setting.

This explains the riddle from the previous subsection. By keeping the `learning_rate` parameter the same when scaling to k times larger effective batch, we actually increase the actual learning rate $\sqrt{k}$ times, in accordance with the suggestion of Hoffer et al. [2017].[30] This holds only for the `linear_warmup_-rsqrt_decay` schedule and ignoring the warmup steps.

If we want to keep the same learning rate also in the warmup phase, we would need to divide the warmup steps by k. However, this means that the maximum actual learning rate will be $\sqrt{k}$ times higher, relative to the single-GPU maximal actual learning rate and this leads to divergence in our experiments. Indeed, many researchers [e.g. Goyal et al., 2017] suggest to use a warmup when scaling to more GPUs in order to prevent divergence. Transformer uses learning rate warmup by default even for single-GPU training (cf. Section 4.3.6), but it makes sense to use more warmup training examples in multi-GPU setting.

In our experiments with 8 GPUs and the default learning rate 0.20, using 8k warmup steps instead of the default 16k had no effect on the BLEU curve (it was slightly higher in the first few hours, but the same afterwards). Further decreasing the number of warmup steps resulted in a very slow BLEU convergence (in the case of 6k warmup steps) or a complete divergence (in the case of 2k warmup steps).

[28] Examples of learning rate schedules are inverse-square-root decay, inverse-time decay, exponential decay or piecewise-constant decay, see https://www.tensorflow.org/api_guides/python/train#Decaying_the_learning_rate for TF implementations.

[29] By *actual* learning rate, we mean the learning rate after applying the decay schedule. The `learning_rate` parameter stays the same in this case.

[30] In addition to suggesting the $\sqrt{k}$ learning-rate scaling, Hoffer et al. [2017] show that to fully close the "generalization gap", we need to train longer because the absolute number of steps (updates) matters. So from this point of view, using steps instead of epochs as the time parameter for learning rate schedules may not be a completely wrong idea.

Conclusions Related to Learning Rate and Warmup Steps on Multiple GPUs

- It is recommended to keep the `learning_rate` parameter at its optimal value found in single-GPU experiments.
- The number of warmup steps can be decreased but less than linearly and without expecting to improve the final BLEU this way.

4.3.9 Resumed Training

T2T allows to resume training from a checkpoint, simply by pointing the `output_dir` parameter to a directory with an existing checkpoint (specified in the `checkpoint` file). This may be useful when the training fails (e.g. because of hardware error), when we need to continue training on a different machine or during hyper-parameter search (when we want to continue with the most promising setups). T2T saves also Adam momentum into the checkpoint, so the training continues almost as if it had not been stopped. However, it does not store the position in the training data – it starts from a random position. Also the relative time (and wall-clock time) in TensorBoard graphs will be influenced by the stopping.

Resumed training can also be exploited for changing some hyper-parameters which cannot be meta-parametrized by the number of steps. For example, Smith et al. [2017] suggest to increase the effective batch size (and number of GPUs) during training, instead of decaying the learning rate.

Yet another usage is to do domain adaptation by so-called *fine-tuning* – by switching from (large) general-domain training data to (small) target-domain training data for the few last epochs (cf. Section 5.1). In this case, we should consider tuning also the learning rate or learning rate schedule (or faking the `global_step` stored in the checkpoint) to make sure the learning rate is not too small.

4.3.10 Checkpoint Averaging

Vaswani et al. [2017] average all the weights in the last 20 checkpoints saved in 10-minute intervals (see Section 5.2.2). According to our experiments, slightly better results are achieved with averaging checkpoints saved in 1-hour intervals.[31] This has also the advantage that less time is spent with checkpoint saving, so the training is faster. We have implemented a tool `t2t-avg-all` for automatic checkpoint averaging, which can be combined with our scripts `t2t-translate-all` and `t2t-bleu` for continuous checkpoint evaluation.[32]

Figure 4.10 shows the effect of averaging is twofold: the averaged curve has a lower variance (flickering) from checkpoint to checkpoint and it is almost

[31] We carried out these experiments both on a single GPU and on 8 GPUs. With more than 8 GPUs, the optimal interval (and total training time) may be lower. In practice, also the available disk space must be considered.

[32] All three scripts are available at `https://github.com/tensorflow/tensor2tensor/tree/master/tensor2tensor/bin/`.

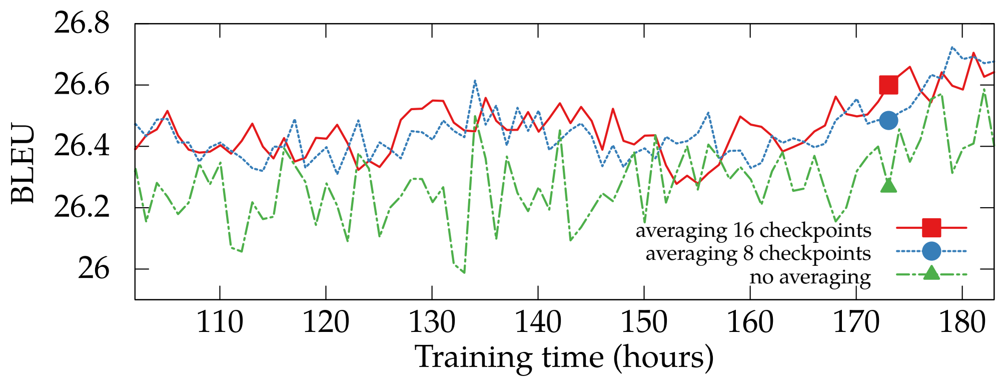


Figure 4.10: Effect of checkpoint averaging. All trained on 6 GPUs.

always better than the baseline without averaging (usually by ca. 0.2 BLEU). In the early phases of training, while the (baseline) learning curve grows fast, it is better to use fewer checkpoints for averaging. In later phases (as shown in Figure 4.10, after 4.5–7.5 days of training), it seems that 16 checkpoints (covering last 16 hours) give slightly better results on average than 8 checkpoints, but we have not done any proper evaluation for significance (using paired bootstrap testing for each hour and then summarizing the results).

The fact that resumed training starts from a random position in the training data (cf. Section 4.3.9) can be actually exploited for "forking" a training to get two (or more) copies of the model, which are trained for the same number of steps, but independently in the later stages and thus ending with different weights saved in the final checkpoint. These semi-independent models can be averaged in the same way as checkpoints from the same run, as described above. In our experiments using three "forks" helped ca. 0.1–0.2 BLEU relative to the default (no-fork) checkpoint averaging (the forks were done after 1, 2 and 4 days of training on a single GPU, the final evaluation was done after 9 days after the first fork). However, in further experiments we decided to use multi-GPU setups instead and we had not enough computational resources to try forking with multiple GPUs.

Conclusions Related to Checkpoint Averaging

- Averaging 8 checkpoints takes ca. 5 minutes, so it is a "BLEU boost for free" (compared with the time needed for the whole training).[33]
- See also Chapter 5 for research on the synergy of checkpoint averaging with backtranslation.

[33] The averaging was done on CPU (Intel Xeon E5-2620). The speed depends mostly on the size of the checkpoints (2.4GB in our case).

4.4 Comparison with WMT2017 Systems

Table 4.6 provides the results of WMT2017 English→Czech news translation task, with our best Transformer model (BIG trained on 8 GPUs for 8 days, averaging 8 checkpoints) evaluated on the `wmt17` test set using exactly the same implementation of automatic metrics. See Section 2.2.2 for details on the evaluation.

	automatic			manual	
system	BLEU cased	BLEU uncased	chrF2 cased	refDA Avg %	refDA Ave z
our Transformer	**23.84**	**24.40**	**0.5164**		
Nematus (UEdin)	22.80	23.29	0.5059	62.0	0.308
Google	20.12	20.57	0.4847	59.7	0.240
LIMSI-factored	20.22	20.92	0.4870	55.9	0.111
LIUM-FNMT	19.95	20.69	0.4847	55.2	0.102
LIUM-NMT	20.20	20.61	0.4841	55.2	0.090
Chimera (CUNI)	20.51	21.65	0.4834	54.1	0.050
Moses (JHU)	19.12	19.64	0.4760		
Bing	16.55	17.55	0.4593	53.3	0.029
PJATK	16.18	16.32	0.4302	41.9	−0.327
TectoMT (CUNI)	12.91	13.43	0.4346		

Table 4.6: Evaluation of WMT2017 systems and our Transformer model on the `wmt17` test set. Human evaluation scores are taken from the official ranking [Bojar et al., 2017a].

The best English→Czech system in WMT2017 according to both automatic and manual evaluation was Nematus (UEdin) [Sennrich et al., 2017]. It is an ensemble of eight Deep-RNN models (4 left-to-right and 4 right-to-left) trained with back-translated data (see the next chapter). Our Transformer model (in this chapter) does not use any of these techniques and still, it outperforms Nematus by 1 BLEU point (which is significant, $p < 0.05$). In Popel and Bojar [2018], we report Transformer is better than Nematus also in three more automatic metrics.

In the next chapter, we present further improvements, evaluated again on `wmt17` in Section 5.4. We acknowledge the automatic evaluation is not fully reliable – in Chapter 6, we present the final evaluation on `wmt18` including manual evaluation.

4.5 Conclusions

We presented a broad range of experiments with the Transformer model [Vaswani et al., 2017] for English→Czech neural machine translation. While we limited our exploration to the standard hyper-parameters provided in the T2T framework, we observed interesting phenomena, some of which are novel to the best of our knowledge. We believe this chapter can be useful for other researchers. In total, all experiments done for this chapter took approximately 4 years of GPU time.

The most notable of our observations about the Transformer model implemented in the Tensor2Tensor framework are:

- Larger batch sizes lead not only to faster training, but more importantly better translation quality.
- For training longer than one day on 11GB GPU, the larger setup (BIG) should be always preferred.
- Using as many GPUs as possible and running experiments one after another should be preferred over running several single-GPU experiments concurrently.
- The best performing model we obtained on 8 GPUs trained for 8 days has outperformed the WMT2017 winner according to BLEU and chrF2.

Chapter 5

Backtranslation and Checkpoint Averaging in NMT

Essentially, all models are wrong, but some are useful.
George Box

This chapter presents improvements of the NMT system described in the previous chapter. The improvements in translation quality (+3 BLEU for English →Czech) are achieved by exploiting monolingual data according to the current best practices (+1 BLEU) and by improving these practices with novel techniques (+2 BLEU). The synergy of *concat backtranslation* and *checkpoint averaging* described in Sections 5.3.2 and 5.3.3 is one of the key achievements of the whole thesis with both theoretical (cf. the discussion in Section 5.5) and practical (cf. the evaluation in Section 5.4 and Chapter 6) impact.

5.1 Introduction

The quality of NMT depends heavily on the amount and quality of the training parallel sentences. Although millions of parallel sentences are freely available for several language pairs,[1] it is still not enough for achieving optimal results. Moreover, the parallel data may not be available for the target domain (e.g. medical or IT domain). A common solution is to use monolingual target-language data, which is usually available in much larger amounts than the parallel data. The current best practice in improving the quality of NMT using monolingual target-language data is *backtranslation* [Sennrich et al., 2016a], where the monolingual data is machine-translated to the source language, and the resulting sentence pairs are used as additional (*synthetic*) parallel training data.[2]

Sennrich et al. [2017] compared two regimes of how to incorporate the synthetic training data. In the *fine-tuned* regime, a system is trained first on the

[1] `http://opus.nlpl.eu/` provides 32 English-X parallel corpora larger than 500M tokens for the following 17 languages: ar, bg, cs, de, el, es, fr, hr, hu, it, nl, pl, pt, ro, ru, sr, tr.

[2] In translation studies (translatology), "backtranslation" means *translating a previously translated document back into the original language*. However, in this thesis we follow the NMT terminology of Sennrich et al. [2016a], where "backtranslation" means that for a final X→Y MT system, we prepare (*back-translate*) additional training data using an MT system in the opposite direction, i.e. Y→X. This data is almost always originally written in the language Y (not translated to Y), thus the *NMT backtranslation* does not overlap with the *translation-studies backtranslation*.

authentic parallel data and then after several epochs it is trained on a 1:1 mix of authentic and synthetic data. In the *mixed* regime, the 1:1 mixed data is used from the beginning of training. In both cases, the 1:1 mix means shuffling the data randomly at the sentence level, possibly oversampling the smaller of the two data sources.

In this chapter, we introduce a third approach, termed *concat* regime, where the authentic and synthetic parallel data are simply concatenated (without shuffling). We observe that this regime changes the training dynamics and leads to improvements in translation quality relative to both *mixed* and *fine-tuned* regimes (Section 5.3.2).

We also further explore the effect of *checkpoint averaging* (averaging all weights in the last N checkpoints, i.e. model snapshot files saved e.g. each hour), which is a light-weight alternative to checkpoint ensembling. We observe surprising synergic improvements (Section 5.3.3) when using the checkpoint averaging together with the concat-regime backtranslation. We explore these improvements (Sections 5.3.4–5.3.8) and analyze the differences in translation outputs from the various models (Section 5.3.9). Finally, we provide suggestions for further improvements: filtering the training data (Section 5.3.10) and building two separate models based on the original language of the text to be translated (Section 5.3.11).

5.2 Related Work

5.2.1 Backtranslation

In SMT, target-language monolingual data are usually used only to build language models, but there are also works showing that additional synthetic parallel training data created with backtranslation can improve SMT [Schwenk, 2008, Bertoldi and Federico, 2009, Bojar and Tamchyna, 2011].

The early end-to-end NMT systems (e.g. Bahdanau et al. [2014]; see Section 2.5) used no target-language monolingual data. First attempts to exploit this abundant data to improve NMT used a separately-trained RNN language model [Gülçehre et al., 2015] or sentence pairs with dummy (empty) source side used to train only the NMT decoder [Sennrich et al., 2016a]. However, synthetic parallel data created by backtranslation quickly became popular because it is very easy to use with any NMT architecture and it gives better results than both aforementioned approaches [Sennrich et al., 2016a]. So while in SMT, target-language monolingual data are typically used for building language models, in NMT they are typically used for building synthetic parallel data via backtranslation.

Note that although the data is termed *synthetic*, it is only its source side which is machine-translated. The target side is authentic and thus can improve the fluency (and sometimes even adequacy) of the final translations, simply by increasing the total size and diversity of the training data.[3] Backtranslation can also be used as a domain-adaptation technique if the target-language monolingual data is in-domain or filtered to match the target domain [Moore and Lewis, 2010].

Poncelas et al. [2018] explore the optimal ratio of authentic and synthetic training data. According to their German→English experiments, using 1M authentic + 1M synthetic sentences gives better results than using 2M authentic sentences. Even better results are achieved with 1M authentic and 2M synthetic sentences and the authors suggest that 1:2 is the optimal authentic-to-synthetic ratio for the given dataset because adding more synthetic data (2.5M sentences) resulted in slightly worse results. However, they have not tried oversampling the authentic data, so we consider the results inconclusive.[4]

5.2.2 Checkpoint Averaging

A popular way of improving the translation quality in NMT is ensembling, where several independent models are trained and during inference (decoding) each target token is chosen according to an averaged probability distribution (using argmax in the case of greedy decoding) and used for further decisions in the autoregressive decoder of each model.

However, ensembling is expensive both in training and inference time. The training time can be decreased by using *checkpoint ensembles* [Sennrich et al., 2017], where N last checkpoints of a single training run are used instead of N independently trained models. Checkpoint ensembles are usually worse than independent ensembles [Sennrich et al., 2017], but allow to use more models in the ensemble thanks to shorter training time. The inference time can be decreased by using *checkpoint averaging*, where the weights in the N last checkpoints are element-wise averaged, creating a single averaged model.

Averaging weights of independently trained models (with different random initialization, including embedding weights) does not work because the model weights are not "compatible". Utans [1996] suggests to train an initial network on all data and then use it as a starting point for training N networks on different subsets of the data and subsequently average weights of these networks. So similarly to checkpoint averaging, the averaged models share the same initial training and are not completely independent.[5]

Checkpoint averaging has been first used in NMT by Junczys-Dowmunt et al. [2016, § 6.3], who report that averaging four checkpoints is "*not much worse than the actual ensemble*" of the same four checkpoints and it is better than ensembles

[3] Rarrick et al. [2011] show that it is beneficial to filter out machine-translated sentences from the training data, but this concerns target-side synthetic sentences, so it does not contradict the improvements caused by backtranslation.

[4] Poncelas et al. [2018] used 15 training epochs for all experiments, without reporting the full learning curves nor any indication of convergence. Their best result was achieved with 3M authentic sentences, which was surprisingly even better than 3.5M authentic sentences.

[5] In Section 4.3.10, we show that this technique of N "forked" networks can be combined with checkpoint averaging.

of two checkpoints. Averaging ten checkpoints "*even slightly outperforms the real four-model ensemble*".

Checkpoint averaging is popular in recent NMT systems [e.g. Vaswani et al., 2017] because it has almost no additional cost (averaging takes a few minutes), the results of averaged models have lower variance in BLEU and are almost always at least slightly better than without averaging (Section 4.3.10).

Checkpoint averaging and its interplay with training dynamics is still not fully explored and understood. An example of recent surprisingly promising improvements is *stochastic weight averaging* [Izmailov et al., 2018], where checkpoint averaging is used in combination with *cyclic learning rate* [Loshchilov and Hutter, 2016] or constant learning rate and it leads to faster convergence and better generalization, as it finds much broader optima than standard SGD.

We hypothesize that both our concat-regime backtranslation and *stochastic weight averaging* exploit a similar mechanism of supplementary diversification of the checkpoints being averaged (in addition to the diversity caused by conventional training with decaying learning rate).

5.3 Experiments

In our experiments, we use the Transformer BIG model [Vaswani et al., 2017] as described in the previous chapter, except for optimizing our models with the Adafactor algorithm [Shazeer and Stern, 2018] instead of the default Adam.[6] We store checkpoints each hour and train on four or eight GTX 1080 Ti GPUs. We evaluate translation quality with BLEU on the `wmt13` test set (unless stated otherwise) as described in Section 2.2.2.

Our training data is constrained to the data allowed in the WMT2018 shared task.[7] Authentic parallel data (`auth`) are: CzEng 1.7, Europarl v7, News Commentary v11 and Common Crawl. See Section 2.1 for details on data sizes. The synthetic parallel data were created by translating Czech News Crawl articles to English using Nematus 2016 [Sennrich et al., 2016c] and filtering out ca. 3% of sentences (see Section 5.3.10). We split the synthetic data (`synth`) into two parts: articles from 2007–2016 (57M sentences, `synth07-16`) and articles from 2017 (7M sentences, `synth17`).

Note that usually the amount of available monolingual data is orders of magnitude larger than the available parallel data, but in our case it is comparable (58M `auth` vs. 64M `synth`).

[6] We use T2T version 1.6.0, `transformer_big` model and hyper-parameters `batch_size=2900, max_length=150, layer_prepostprocess_dropout=0, learning_rate_schedule=rsqrt_decay, learning_rate_warmup_steps=8000, optimizer=Adafactor`. For decoding, we use `alpha=1.0`.

[7] We use almost all the available parallel English-Czech and monolingual Czech training data allowed for the WMT constrained task. We exclude only the ParaCrawl corpus (`https://paracrawl.eu/`) because we found even the filtered 10M en-cs sentences too noisy.

5.3.1 Training on Fully Shuffled Data

Figure 5.1 shows that relative to the authentic-only data, training on mixed (authentic and synthetic) data results in faster convergence (higher BLEU in the early training), but only a small improvement (0.1–0.2 BLEU) after 50 hours of training on 4 GPUs. This contrasts with Sennrich et al. [2016a] reporting an 2.8–3.4 BLEU improvement on English-German by using 3.6M synthetic sentences in addition to 4.2M authentic sentences. Note that our English-Czech data is more than ten times larger, so it is expected that backtranslation as a data-augmentation technique will lead to smaller improvements than on the English-German pair. In Figure 5.1, one training epoch takes about 11 hours for the `auth` data (58M sentences), 16 hours for `synth` (64M sentences) and 27 hours for `mix` (122M sentences).

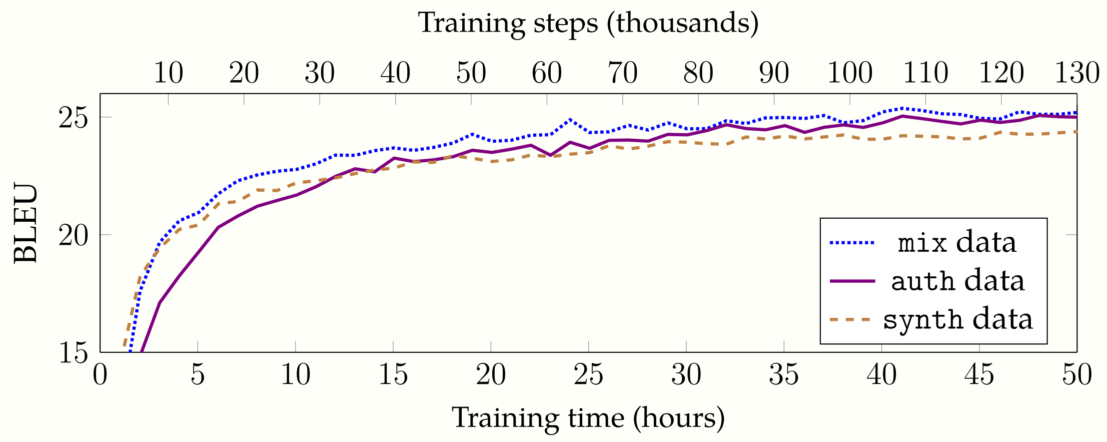


Figure 5.1: Learning curves for training on authentic, synthetic and mixed data. All trained on 4 GPUs.

Training on `synth` leads to the highest BLEU in the first two hours of training, but it is 0.6 BLEU worse than `auth` after 50 hours. The source side (English, in this case) of `synth` is created by an MT system, so we hypothesize it is more regular in some sense than the authentic data and thus easier for the encoder to exploit the regularities.

Our experiments (data not shown) with *fine-tuned* regime (first train on `auth`, then switch to `mix`) led to the same final BLEU as training on `mix` from the beginning, in accordance with the results of Sennrich et al. [2016a]. Switching from `auth` to `mix` at different moments (20, 30 and 40 hours) did not markedly improve the final BLEU after 50 hours of training (data not shown).

5.3.2 Concat Regime Training

Each of the three training datasets (`auth`, `synth` and `mix`) in Figure 5.1 was shuffled on the sentence level. Now, we contrast it with a `concat` dataset, created simply by concatenating: `auth`, `synth07-16`, `auth` and `synth17`, in this order. The sentences *within* these four data blocks are still shuffled, we only do not shuffle

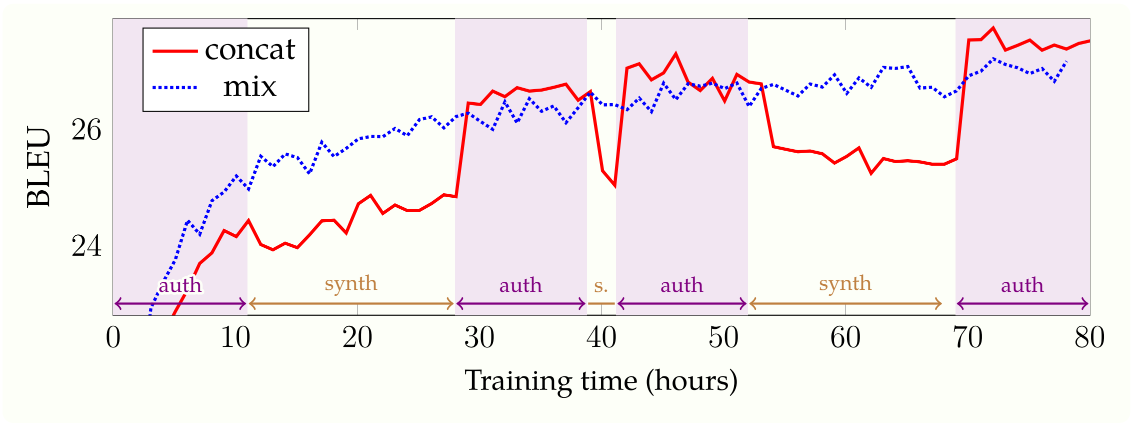


Figure 5.2: Learning curves for training on *concat* vs. *mixed* regime backtranslation. Both trained on 8 GPUs. Arrows show `auth/synth` training spans for *concat*.

sentences *across* the data blocks. Note that we have included `auth` twice in the concatenation, so the total ratio of authentic:synthetic data is 116:64.[8] Another motivation was to explore a bit more the effect of the size of concatenated data blocks (cf. Section 5.3.4).

For further experiments, we switch to 8 GPUs, in order to have a stronger baseline [cf. Denkowski and Neubig, 2017]. Note that this doubles the *effective batch size* relative to the 4GPU experiments, which has a positive effect on BLEU in addition to faster training (see Section 4.3.7).

Figure 5.2 shows that when the model is being trained on `auth`, the dev-set BLEU suddenly increases by ca. 1.6 BLEU. In these `auth`-training phases, the model is on average ca. 0.3 BLEU better than when training on `mix`.

5.3.3 Checkpoint Averaging Synergy

The thick and thin blue curves in Figure 5.3 show that, as expected, averaging the last 8 checkpoints has a small positive effect (+0.3 BLEU on average) when training on `mix`.[9] Surprisingly, the positive effect on `concat` (red curves) is much larger (over +1.3 BLEU). Importantly, by comparing the thick curves (taking the maximal value achieved at a given time), we see that averaged `concat` is over 1.1 BLUE better than averaged `mix`.

We continued training up to 9 days (1M steps), where the best dev-set BLEU climbs up to 28.8 for `concat-avg8` and 27.8 for `mix-avg8`,[10] while the curves

[8] For further experiments in this chapter, we have oversampled authentic data twice also in the `mix` dataset, so it has the same size as `concat` – 180M sentences.

[9] When training longer than 80 hours, the improvement caused by averaging on `mix` is slightly smaller – ca. +0.2 BLEU (in accordance with our observations in Section 4.3.10).

[10] The optimal BLEU=27.80 of `mix-avg8` was achieved at 607k steps, then it became slightly decreasing (possibly overfitting the training data). For `concat-avg8`, the optimal BLEU=28.84 was achieved at a 775k-steps peak and further peaks at 971k and 1168k steps achieved almost the same BLEU. We concluded that no substantial improvement could be achieved by further training.

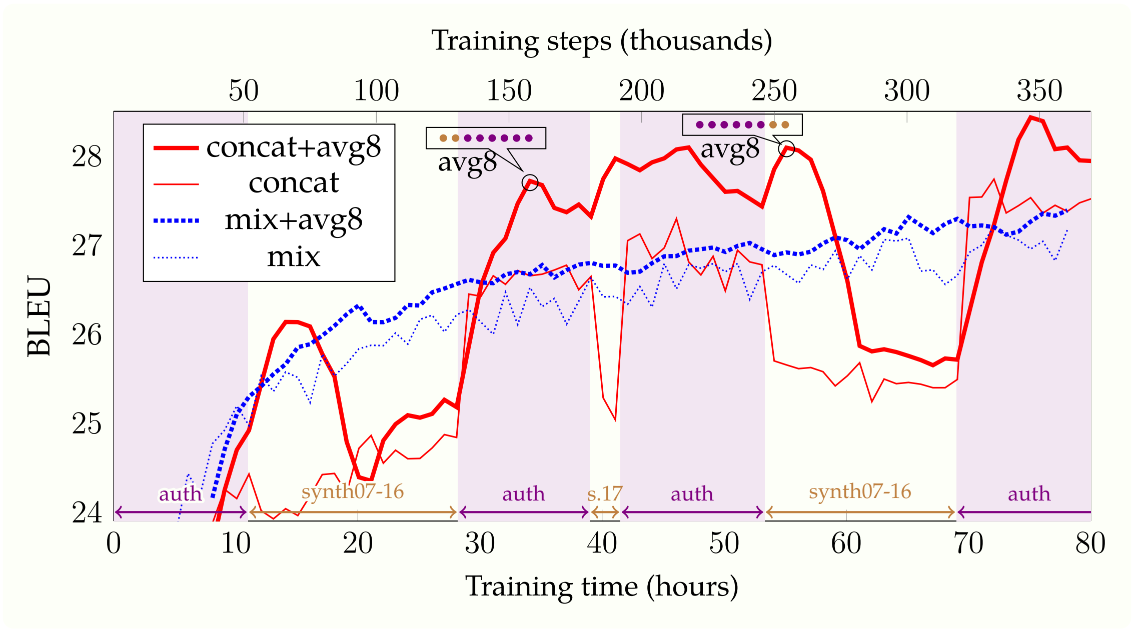


Figure 5.3: The effect of averaging 8 last checkpoints with *concat* and *mixed* regime backtranslation. The callouts (pointing to the "initial" and "final" peaks of the concat+avg8 curve) illustrate the training datasets of the 8 averaged checkpoints (synth as brown circles, auth as violet circles). Note that both peaks have the same ratio of 6 `auth`-trained and 2 `synth`-trained checkpoints in the average.

follow still the same pattern. Evaluation on our test set (`wmt17`; see Table 5.1) confirms that the improvement of `concat-avg8` relative to `mix-avg8` is ca. +1.0 BLEU even for these fully trained models.

The "initial" peak of `concat-avg8` (at 34 and 75 hours in Figure 5.3) follows six hours after starting training on `auth` and the "final" peak (at 55 hours in Figure 5.3) follows two hours after starting training back on `synth07-16`. Note that in both cases, the optimal ratio of high-BLEU (`auth`-trained) and low-BLEU (`synth`-trained) checkpoints in the average is 6:2.[11]

5.3.4 Data Block Size Effect

We increased the size of the authentic data block by repeating `auth` three times (data not shown). The maximal improvement caused by averaging was again ca. +1.3 BLEU, six hours after switching to `auth`. When training longer on `auth`, the improvement caused by averaging gradually decreased to ca. +0.3 BLEU (similarly as when training on `mix`).

In another experiment, we decreased `auth` size, so training on it took only three hours (data not shown). The 8-checkpoint averaging curve reached its maximum at the auth:synth ratio of 3:5, which was the highest ratio achievable

[11] In further training (data not shown), sometimes the optimal ratio is 5:3, as the auth/synth switch is not exactly aligned to the one-hour checkpoint interval.

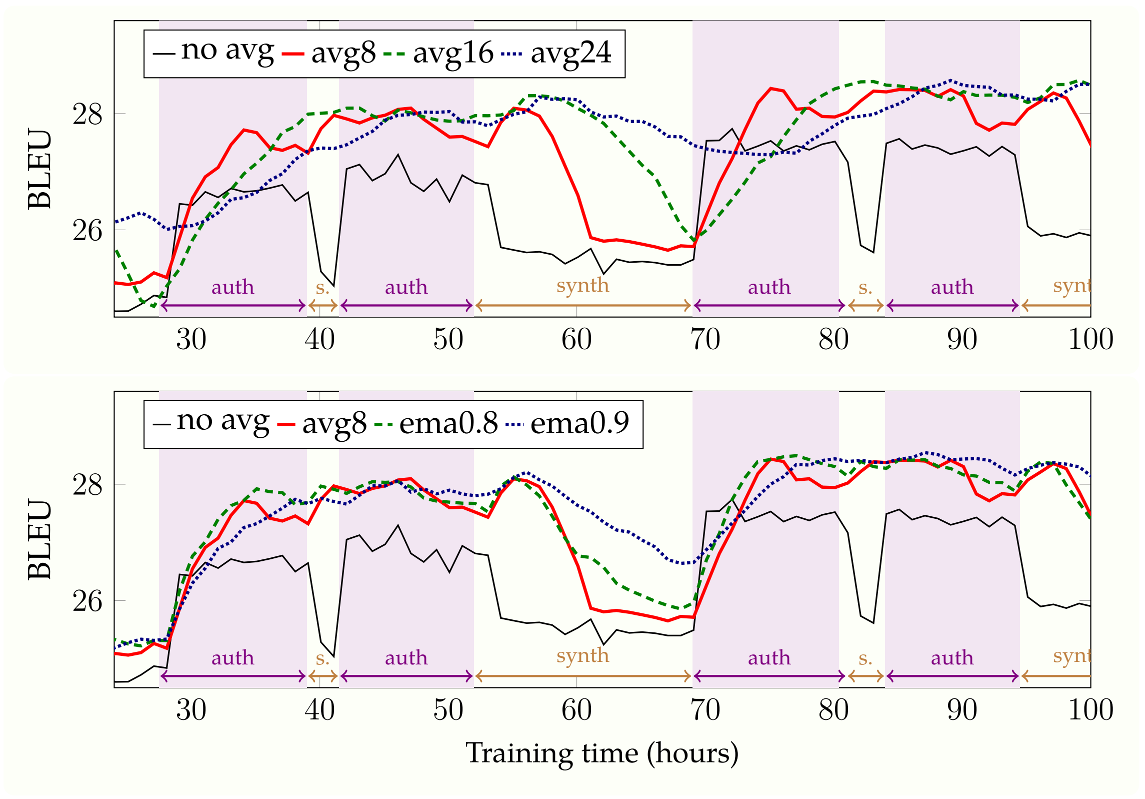


Figure 5.4: Effect of averaging 8, 16 and 24 checkpoints and exponential moving averaging (ema) with interpolation constants 0.8 and 0.9.

in this setting, but the BLEU was lower than in Figure 5.3 because the ratio was far from its optimal value of 6:2.

The advantage of concat-regime backtranslation with checkpoint averaging is that we do not need to know the optimal ratio of `auth` and `synth` in advance (unlike in mixing). We only need to keep both of these data blocks large enough (possibly by oversampling, depending on the chosen number of checkpoints to average), so that all ratios are explored during training and evaluated on the development set.

5.3.5 Training History Effect

We took the three models from Figure 5.1 (trained on `auth`, `synth` and `mix`) and continued training on `concat`. In all three cases, we obtained similar improvements and final results as when we trained on `concat` from the beginning. We hypothesize the positive effect of concat backtranslation with checkpoint averaging is independent of the way how a given model was trained in the past (unless the training diverged, of course).

5.3.6 Averaging Variants

In addition to 8 checkpoints, we tried averaging also 16 and 24 checkpoints; see Figure 5.4. The optimal ratio was similar (ca. 12:4 and 18:6, respectively), but the peaks were flatter and the effects of auth→synth and synth→auth switch were overlapping. We tried also *exponential moving averages* with interpolation constants 0.8 and 0.9, which led to similar curves as avg16 and avg24, respectively. All these four additional experiments had a similar maximal BLEU as avg8.

5.3.7 Effect of Backtranslation Quality

We tried similar experiments using the Czech→English direction and obtained similar results and relative improvements caused by concat backtranslation. The synthetic data for these experiments were based on English News Crawl articles from 2016–2017, translated by our English→Czech Transformer, which already uses concat backtranslation (as described in Section 5.3.3).

As a "byproduct," we obtain a high-quality Czech→English Transformer system, which we used for translating Czech monolingual News Crawl; however, due to lack of GPU resources, we translated only articles from years 2016 and 2017. We replaced the `synth07-16` and `synth17` data sets translated by Nematus 2016 by `synth16-17` translated by our Transformer with a notably higher translation quality. Our Czech→English Transformer is 5.7 BLEU better than Nematus 2016 on the `wmt16` test set. Out of this difference, 2.5 BLEU is caused by the concat backtranslation and the rest (2.7 BLEU) can be attributed to the higher quality of Transformer relative to Nematus 2016.

We observed a +0.8 BLEU improvement on the final English→Czech translation, after we started training using the concat backtranslation on the higher-quality new data (`synth16-17`). See Table 5.1 for evaluation on the `wmt17` test set.

Note that for obtaining this final English→Czech system (which won the WMT2018 shared task, cf. Chapter 6), we iterated the backtranslation process:

1. We reused the Nematus 2016 models trained by Sennrich et al. [2016c] using *fine-tuned* backtranslation of English News Crawl 2015 articles, which were translated "*with an earlier NMT model trained on WMT15 data*" [Sennrich et al., 2016c].

2. We used Nematus 2016 to translate Czech News Crawl 2007–2017 articles to English.

3. We trained an English→Czech Transformer on this data (`synth07-16` and `synth17`) using concat backtranslation with averaging as described in Section 5.3.3.

4. We used this Transformer model to translate English News Crawl 2016–2017 articles into Czech.

5. We trained our Czech→English Transformer model (mentioned at the beginning of this section) on this data using concat backtranslation with averaging.

6. We translated Czech News Crawl 2016–2017 articles into English using this system, producing (a higher-quality) `synth16-17`.

7. We trained our final English→Czech system on this data, again using concat backtranslation.

5.3.8 Generalization to Other Test Sets

In order to benefit from the described BLEU improvements in practice, we need to test whether they generalize to unseen test sets. For this purpose, we concatenated WMT test sets from 2008–2012 (13.5k sentences total) and observed the same cyclic pattern as in Figure 5.3, with similar relative BLEU improvements, and most importantly with the same optimal ratio 6:2 of `auth`-trained and `synth`-trained checkpoints in the average. However, when testing on `wmt16`, the optimal ratio was 5:3 or even 4:4 and using the models averaged with the 6:2 ratio resulted in lower improvements (+0.4 BLEU) relative to the `mix+avg8` baseline (while the 5:3 ratio results in ca. +1.0 BLEU improvements).

This seemingly disappointing observation led us to further improvements and, more importantly, to a better understanding of the mechanisms behind the observed effects, as described below.

5.3.9 Inspecting the Translation Differences

We inspected the actual translations of our dev set (`wmt13`) and focused on comparing the outputs of a (low-BLEU) `synth`-trained model and a (high-BLEU) `auth`-trained model, both without averaging. We used the MT-ComparEval toolkit [Klejch et al., 2015] for the manual inspection.

On the one hand, we noticed sentences where the `synth`-trained model output was more fluent. For example, "Tories" was kept untranslated in the `auth`-trained output, and correctly/fluently translated to Czech as "Toryové" in the `synth`-trained output. On the other hand, we noticed sentences where the `synth`-trained was less adequate. For example, "rakfisk" was correctly kept untranslated in the `auth`-trained output (as there is no established Czech translation for this Norwegian dish), but incorrectly (though fluently) translated as "rakytník" (meaning *Hippophae* or *sea buckthorn*) in the `synth`-trained output.

We saw a similar type of differences when comparing the outputs of the `synth`-trained and `auth`-trained averaged models, but here the `auth`-trained avg8 output seemed to combine the strengths of both `synth`- and `auth`-trained model outputs. We also observed that the `synth`-trained output sentences are slightly longer on average and include a higher percentage of untranslated sentences (but in our final experiments this is not true – see the next section).

5.3.10 Training Data Filtering

In our initial experiments mentioned in the previous section, ca. 0.5% of dev-set sentences remained untranslated in the `synth`-trained output. We found out that the Czech monolingual data set (News Crawl 2007–2017) contains many English sentences. Those sentences were either kept untranslated or paraphrased when preparing the `synth` data with backtranslation. Thus the `synth` data included many English-English sentence pairs and models trained on `synth` thus had a higher probability of keeping a sentence untranslated.

In order to filter out the English sentences from the Czech data, we kept only sentences containing at least one accented character.[12] We also filtered out sentences longer than 500 characters from the `synth` data. Most of these sentences would be ignored anyway because we are training our Transformer with `max_length`=150, i.e. filtering out sentences longer than 150 subwords (cf. Section 4.3.4). Sometimes a Czech sentence was much shorter than its English translation (especially for the translations by Nematus 2016) – because of filler words repeated many times, which is a well-known problem of NMT systems [Sudarikov et al., 2016]. We filtered out all sentences with a word (or a pair of words) repeated more than twice using a regular expression `' (\S+ ?\S+) \1 \1 '`.

This way, we filtered out ca. 3% of sentences and re-trained our systems. After this filtering, we did not observe any untranslated sentences in the `synth`-trained output. We re-ran all the experiments; all figures and BLEU scores presented in this chapter are thus based on the filtered training data.

5.3.11 Separating Models for CZ and nonCZ

In the `wmt13` test set, 17% of sentences originate from Czech news servers (e.g. aktualne.cz) and the rest from non-Czech news servers (e.g. bbc.com). All WMT test sets include the server name for each document in metadata, so we were able to split `wmt13` (and later other WMT test sets) into two parts: `CZ` (for Czech-domain articles, i.e. documents with `docid` containing ".cz") and `nonCZ` (for non-Czech-domain articles).

Figure 5.5 compares BLEU learning curves of a single concat-backtranslation-trained model evaluated separately on these two parts. When focusing on the thin curves of models without averaging, we can see that when the model is being trained on `synth`, it performs much better on the `CZ` test set than on the `nonCZ` test set. When trained on `auth`, it is the other way round. Intuitively, this makes sense: The target side of `synth` are original Czech sentences from Czech newspapers, similarly to the `CZ` test set. In `auth`, over 90% of sentences were originally written in English about "non-Czech topics" and translated into Czech (by human translators), similarly to the `nonCZ` test set. In Section 5.5.3, we will discuss these two closely related phenomena: a question of domain (topics) in the training data and a question of so-called *translationese* effect, i.e. which

[12] This simple heuristics is surprisingly effective for Czech. In addition to English sentences, it filters out also *some* short Czech sentences, sentences in other languages (e.g. Chinese) and various "non-linguistic" content, such as lists of football or stock-market results.

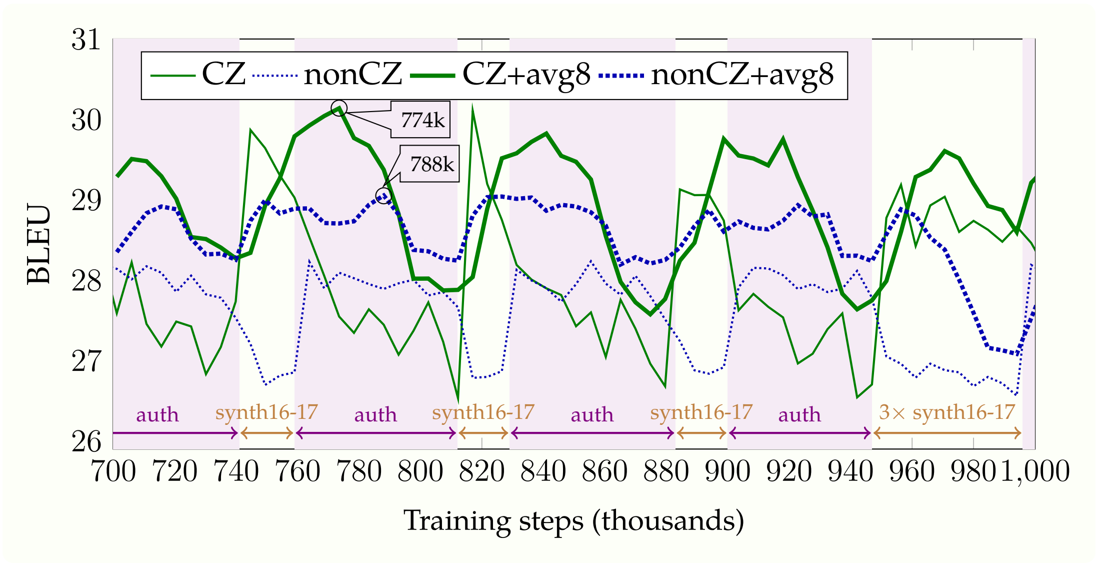


Figure 5.5: Evaluating *concat* backtranslation on the Czech-domain (CZ) and non-Czech domain (nonCZ) portion of `wmt13`. The callouts mark maxima of CZ+avg8 (at 774k steps) and nonCZ+avg8 (at 788k) curves.

side of the parallel training data (and test data) is the original and which is the translation.

In Section 5.3.8, we observed that the optimal ratio of `auth`-trained and `synth`-trained checkpoints differs between `wmt13` and `wmt16` (6:2 vs. 4:4). This can be explained by an interaction of two factors. First, we have shown that the `CZ`-evaluation prefers `synth`-trained models and the `nonCZ`-evaluation prefers `auth`-trained models. Second, `wmt13` contains only 17% of Czech-domain sentences, whereas `wmt16` contains 50% of such sentences. WMT test sets in 2008–2013 were created by selecting news from 5–7 languages and using the same set of news articles for all the language pairs. Thus, the English-Czech test sets in 2008–2013 contain mostly sentence pairs where both sides are actually translations (e.g. an originally Spanish sentences translated to English and Czech). WMT test sets in 2014–2018 use only direct translations for each language pair (so for each language pair X-EN, half of the sentences are originally English and the other half originally X).

Based on these observations, we prepared a CZ-tuned model and a nonCZ-tuned model. Both models were trained and evaluated in the same way as in Figure 5.5. For the CZ-tuned model, we selected the checkpoint with the best performance on `wmt13-CZ` (CZ+avg8 checkpoint at 774k steps, 3:5 ratio). Similarly, for the nonCZ-tuned model, we selected the checkpoint with the best performance on `wmt13-nonCZ` (nonCZ+avg8 checkpoint at 788k steps, 6:2 ratio). Note that both the models were trained jointly in one experiment, just selecting checkpoints at two different moments.

We have confirmed that the model tuned on `wmt13-CZ` obtains near-optimal results on `wmt16-CZ` and similarly for the nonCZ-tuned model and test sets (data not shown). Thus, we have resolved the disappointing generalization problems reported in Section 5.3.8. Moreover, we obtained a further +0.2 BLEU improvement on `wmt17` (Table 5.1) by translating the `wmt17-CZ` portion with the CZ-tuned model and the `wmt17-nonCZ` portion with the nonCZ-tuned model.

5.3.12 Tuning Translation Length

We also tried tuning the beam-search parameter α [Wu et al., 2016], which affects the translation length.[13] For these experiments, we concatenated the WMT test sets 2008–2016 and split them to `wmt08-16-CZ` (Czech domain, 7 400 sentences) and `wmt08-16-nonCZ` (the rest, 17 800 sentences). We define *length ratio* as the total length of all translations produced by the MT system divided by the total length of all reference translations. All lengths are measured in words.

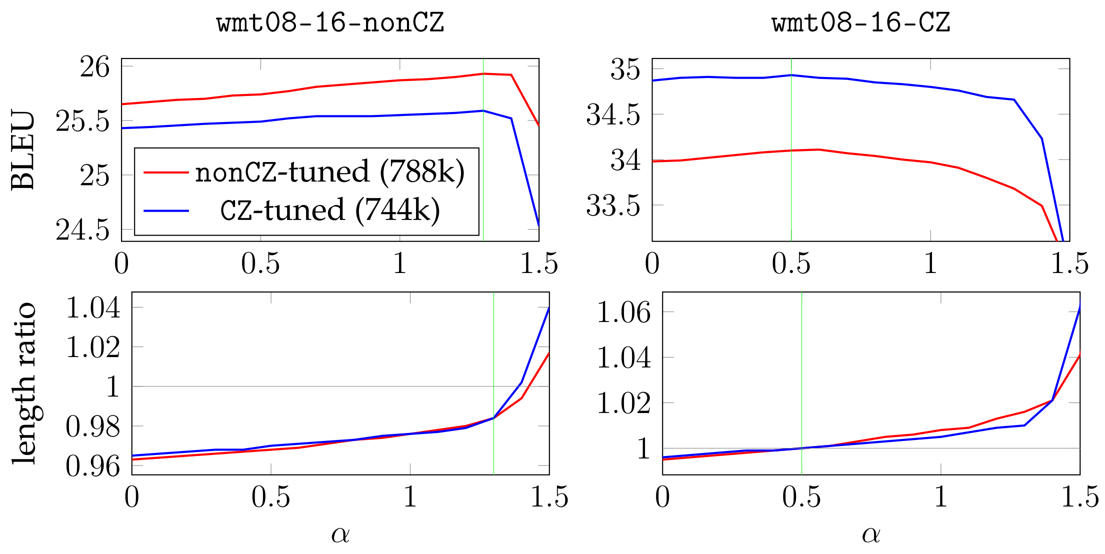


Figure 5.6: Effect of α on BLEU and *length ratio* for the two systems described in Section 5.3.11: `nonCZ`-tuned (788k steps) and `CZ`-tuned (744k steps). The upper graphs show BLEU, the lower graphs show *length ratio*, i.e. length of the translations divided by the length of the references. The left graphs show scores on the `wmt08-16-nonCZ` set and the right graphs on `wmt08-16-CZ`. The green vertical lines mark the α that results in the maximum BLEU.

Figure 5.6 explores the effect of α on our two final systems (`nonCZ`-tuned and `CZ`-tuned) and on these two data sets. The optimal value of α (marked with a green vertical line) according to BLEU is the same for both systems, but different for the two data sets: 1.3 for `wmt08-16-nonCZ` and 0.5 for `wmt08-16-CZ`. However, our default value of α=1.0 (used in all experiments in this chapter) is almost optimal for both the systems. For simplicity, we decided to keep α=1.0 also for our final systems evaluated on `wmt17` (Section 5.4).

[13] In general, lower α leads to shorter sentences in number of subwords. This usually means shorter in number of words. However, sometimes it can lead also the the same number of words, but using more frequent words (which consist of fewer subwords).

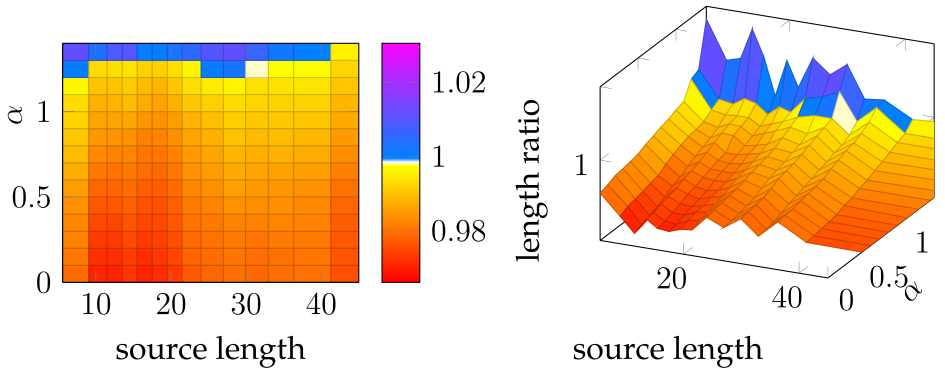


Figure 5.7: Effect of α and source length on the *length ratio* for the `nonCZ`-tuned system evaluated on `wmt08-16-nonCZ`. Red colors mark too short translations, blue colors mark too long translations.

Interestingly, the length ratio is almost the same for the two systems, although the BLEU scores differ. This means that the effect of `CZ`- and `nonCZ`- tuning using concat backtranslation cannot be explained only by tuning the translation length.

Figure 5.6 also shows that for `wmt08-16-nonCZ`, the optimal α (according to BLEU) is achieved with length ratio 0.98, i.e. a bit shorter translations than in the reference translation. Trying to make the translations as long as the reference using the α parameter would result in much lower BLEU (because of lower n-gram precision). In contrast, for `wmt08-16-CZ` the optimal α for both systems is achieved with length ratio 1.0.

Figure 5.7 shows again the effect of α on the *length ratio*, but this time considering also the source-sentence length[14] and focusing on the `nonCZ`-tuned system evaluated on `wmt08-16-nonCZ`. We can see that α makes the translations longer for all source lengths, but it has the largest effect on source lengths of 5–15 words (for which the translations are the shortest relative to reference with α=0, but the longest relative to reference with α=1.4).

5.4 Evaluation on WMT2017

In Table 5.1, we evaluate the models developed in this chapter on an unseen test set – `wmt17` (cf. Section 2.1.2). We report three automatic metrics: case-sensitive (cased) BLEU,[15] case-insensitive (uncased) BLEU[16] and a character-level metric

[14] We bucketed the 17.8k sentence pairs by their source-language length into 15 buckets, each with at least 1000 sentences.

[15] using the default sacreBLEU tokenization for better comparability with the official results at `http://matrix.statmt.org`. The sacreBLEU signature is `BLEU+case.mixed+lang.en-cs+numrefs.1+smooth.exp+test.wmt17+tok.13a`.

[16] using the international tokenization for better comparability with other BLEU results in this thesis. The sacreBLEU signature is `BLEU+case.lc+lang.en-cs+numrefs.1+smooth.exp+test.wmt17+tok.intl`.

system	BLEU cased	BLEU uncased	chrF2 cased
Nematus (WMT2017 winner)	22.80	23.29	0.5059
our baseline Transformer	23.84	24.40	0.5164
our mixed backtranslation	24.85	25.33	0.5267
our concat backtranslation	25.77 (+0.92)	26.29	0.5352
+ higher quality backtranslation	26.60 (+0.83)	27.10	0.5410
+ CZ/nonCZ tuning	**26.81** (+0.21)	**27.30**	**0.5431**

Table 5.1: Final automatic evaluation on the (English→Czech) `wmt17` test set. The three scores in parenthesis show BLEU difference relative to the previous line.

chrF2;[17] see Section 2.2.2 for details. We compare our results with two published systems:

- Nematus [Sennrich et al., 2017] is a Deep-RNN system, which presents the previous state of the art on this test set – it was significantly better than all other systems in WMT2017, also in manual evaluation; see Section 4.4 for details.

- *Our baseline Transformer* is the system presented in Chapter 4 and published in Popel and Bojar [2018]. It is implemented in the T2T framework, trained for 8 days on 8 GPUs with 8-checkpoint averaging, but without any backtranslation.

The most relevant "baseline" for this chapter is:

- *Our mixed backtranslation*, which uses the same setting (including hyper-parameters) as "our baseline Transformer" except that it uses T2T 1.6.0 (instead of 1.2.9 as in Chapter 4) with Adafactor optimizer [Shazeer and Stern, 2018] (instead of Adam) and can thus utilize a higher batch size (2900 instead of 2000) with the same GPU memory. It is trained with mixed-regime backtranslation (which is the previous state-of-the-art technique for exploiting monolingual data).

The test-set results confirm our findings on the dev set:

- Using concat backtranslation (with checkpoint averaging) improves the translation quality by +0.9 BLEU relative to using mixed backtranslation.

- Iterating the process of concat backtranslation and using our Transformer instead of Nematus for producing the synthetic data leads to further improvement of +0.8 BLEU.

- Tuning separate models for translating Czech-domain and non-Czech-domain sentences adds +0.2 BLEU.

[17] `chrF2+case.mixed+lang.en-cs+numchars.6+numrefs.1+space.False+test.wmt17`

All these BLEU improvements are statistically significant ($p < 0.05$), although the reliability of single-reference BLEU for such high (near-human) quality translation is questionable (cf. Section 2.2.2).

In total, our improvements in Chapters 4 and 5 improve the translation quality by 4 BLEU points over the previous (one year old) state of the art. See Chapter 6, for a comparison on `wmt18` with this-year state of the art, including a manual evaluation.

5.5 Discussion

5.5.1 Relation to Best ML Practices

We find it interesting that some of our results and approaches considerably improve the MT performance, even though they seem counter-intuitive at the first sight. For example, when substituting high-BLEU checkpoints in the average with low-BLEU ones, the BLEU of the averaged model is improved (but this can be explained by the complementary strengths of more diverse models). Also, concat-regime backtranslation goes against the standard recommendation that in SGD the training data should be shuffled (or follow the natural order in case of predicting time series, or increasing complexity in case of curriculum learning [Bengio et al., 2009]).

5.5.2 Relation to Dropout

We observed (in an experiment not reported so far) that disabling dropout[18] slightly improves our dev-set BLEU even after several days of training, in contrast with dropout 0.1 and 0.3, which are the values suggested by Vaswani et al. [2017] for EN-FR and EN-DE, respectively. Our EN-CS training data set (58M auth and 64M synth sentences) is much larger than both EN-FR (36M sentences) and EN-DE (4.5M sentences). Dropout is a form of regularization which can be interpreted as an ensemble of exponentially many models with shared weights [Srivastava et al., 2014]. Checkpoint averaging with concat-regime backtranslation can be also viewed as a form of regularization.[19] It is interesting that this regularization is effective even in cases when dropout is not.

5.5.3 Relation to Domain Adaptation and Translationese

Backtranslation is recognized as a form of domain adaptation – Sennrich et al. [2016a] investigated adaptation to multiple domains by *fine-tuned* backtranslation on the respective monolingual data. We show that *concat* backtranslation can be successfully used for joint training of two models – tuned for translation of Czech-origin and non-Czech-origin texts.

[18] setting `layer_prepostprocess_dropout=0`

[19] The training loss when training on `auth` is much higher than on `synth`. Concat backtranslation does not improve the training loss, but it improves the generalization.

There are two closely-related aspects of such domain adaptation. The first aspect is adaptation to the topics of the translated texts. English-origin news usually report about USA, UK or worldwide topics, while Czech-origin news often report about Czech-specific topics (worldwide topics are also covered, but usually from the Czech point of view). The second aspect is adaptation to *translationese* – adaptation of an English→Czech MT system to translation of original-English texts vs. texts translated to English from another language.[20]

We have noticed both these aspects in our experiments, but it is difficult to exactly measure their impact because we do not have enough training parallel texts about Czech topics written originally in English and later translated to Czech.

When comparing the distribution of sentence lengths in our `CZ` and `nonCZ` data, we observed an example of the translationese aspect. The `nonCZ` data set (mostly original-English) contains many short English sentences where the Czech translation is notably longer. Most of these sentences are news headlines, where in English it is common to use a special style (or even grammar) for brevity, e.g. "*Grave-tenders robbed*" translated as "*Návštěvník hřbitova při údržbě hrobu okraden*" (*A visitor of a graveyard robbed during the maintenance of a grave*).

In Section 2.6, we described positive and negative aspects of translationese. Ideally, MT should simulate professional (high-quality) translators, i.e. using translationese only where necessary or appropriate. However, at least 40% of our parallel training data was not translated by professional translators and even the rest may contain untranslated or misaligned segments.

5.5.4 Relation to Unsupervised Machine Translation

Our *iterated backtranslation* described in Section 5.3.7 is similar to the *on-the-fly backtranslation* [Artetxe et al., 2018] used in unsupervised NMT, where both translation directions are trained jointly and "*as training progresses and the model improves, it will produce better synthetic sentence pairs through backtranslation, which will serve to further improve the model in the following iterations*" [Artetxe et al., 2018]. In *on-the-fly backtranslation*, the translation direction being trained is switched after each batch, while in *iterated backtranslation* it is switched after training the full system for several epochs.

[20] If the other language is Czech, we deal with *re-translation*, also known as *backtranslation* in the field of translation studies (cf. footnote 2). Thus, backtranslation in the *NMT* sense is useful for backtranslation in the *translation-studies* sense.

5.6 Conclusions

We have presented three novel methods for improving MT quality:

- *concat-regime* backtranslation, possibly applied iteratively,
- checkpoint averaging of diverse models trained on different types of data, and
- domain adaptation by jointly tuning two models specialized for translation of Czech-origin texts (translationese on the source side) and of non-Czech-origin texts (mostly original English on the source side).

These methods can be used separately,[21] but the strongest synergy is achieved when they are used jointly.

[21] Concat backtranslation without averaging is presented in Figure 5.2. We focus on diversifying the averaged checkpoints by concat backtranslation, but one could imagine many other ways, e.g. switching different domains or genres. Finally, two models (CZ and nonCZ) could be tuned even without concat backtranslation, e.g. by fine-tuning two separate models on authentic parallel original-translationese and translationese-original data, if this metadata is available.

Chapter 6

Final Evaluation and Discussion

The original is unfaithful to the translation.
Jorge Luis Borges

In the previous chapters, we described individual components and aspects of our two MT systems (TectoMT and Transformer). Our evaluations in the respective chapters focused mostly on the impact of these components and aspects. In this chapter, we evaluate each system as a whole and compare it with other systems submitted to WMT each year. In Section 6.1, we present a summary of the state of the art in English→Czech MT over the last twelve years, which witnessed several paradigm shifts. In Section 6.2, we present the results of the WMT2018 automatic and manual evaluation of English→Czech and Czech→English MT systems. We show that our Transformer system was ranked in the manual evaluation as significantly better than the human reference translations. Finally, in Section 6.3, we discuss the surprising result and provide a list of hypotheses.

6.1 WMT English-to-Czech Evaluations Summary

We present a summary of the results of both automatic and manual evaluation of WMT shared tasks in 2007–2018. See Section 2.3 for a description of WMT and the categories of MT systems presented in this section. See Section 2.2 for a description of the automatic and manual evaluation methods. To remind the basic facts: The WMT manual evaluation involves usually over 2000 judgments per a single MT system. The secret test set usually contains ca. 3000 sentences taken from recent news articles.

Table 6.1 presents an overview of six categories of MT systems (as defined in Section 2.3). For each category and each year, the best system according to the manual WMT evaluation was selected. We report both the "well-known" names of MT systems or frameworks (such as Moses or Nematus), as well as the names that are reported in the official WMT results, so the readers of this thesis can find more details about the systems in WMT proceedings.[1] Note that even if a same system name is reported for several years, it almost always represents a different system – trained on newer (and larger) training data. Sometimes even the system architecture and training techniques differ notably, e.g. `CU-Bojar` in 2007–2011 used different configurations of the Moses system.

[1] Available from `http://www.statmt.org/wmt17/results.html` and similarly for other years,

category	system name (as reported in official WMT results)
TectoMT	TectoMT, Chapter 3 (`CU-Popel` in 2011, `CU-TectoMT` other years)
RBMT	PC-Translator (`PCT` in 2007, `PC-Trans` in 2009–2010, `Commercial2` in 2011–2014)
SMT	Moses [Koehn et al., 2007] (`CU-Bojar` in 2007–2011 and 2013, `UEdin` in 2012, `UEdin-unconstrained` in 2014, `UEdin-JHU` in 2015, `JHU-PBMT` in 2016, `Moses` in 2017–2018)
Online	online system (`Google` in 2009, `Online-B` since 2010)
Chimera	Chimera [Bojar et al., 2013], Section 2.3 (`CU-Depfix` in 2012–2014, `CU-Chimera` since 2015)
NMT	various systems: `Montreal` (GroundHog) in 2015 [Jean et al., 2015], `UEdin-NMT` (Nematus) in 2016–2017 [Sennrich et al., 2016c, 2017], `CUNI-Transformer` (T2T, Chapter 5) in 2018

Table 6.1: Overview of systems selected for our summary. Abbreviations used: Charles University (CU, CUNI), University of Edinburgh (UEdin), Johns Hopkins University (JHU). The names of commercial online systems are anonymized in WMT since 2010.

Figure 6.1 shows BLEU scores of the selected six English→Czech systems or categories of systems. In addition, we plot a median BLEU (out of all systems submitted in a given year). Note that there is a different test set each year, so the BLEU scores are not comparable across years. For example, we can see a “drop” in BLEU in years 2012, 2015 and 2017 for all systems, which is likely caused by a more “difficult” test set or the style of reference translations.[2] However, note that the RBMT system submitted to WMT in 2010–2015 is exactly the same version of PC-Translator and its BLEU curve for these years is almost constant within the range 9–11 BLEU, which roughly corresponds to the variance in test set “difficulty”.[3] Taking this into account, we can see the general trend of growing BLEU score over years for the best system.

Table 6.2 and Figure 6.2 show the results of manual evaluation. See Section 2.2.1 for an overview of the evaluation types. As can be seen in Table 6.2, the range of manual evaluation scores varies greatly over the years, so in Figure 6.2, we plot a relative score for each system: we scale the scores linearly into interval $\langle 0, 1 \rangle$ (the worst system gets 0 and the best system gets 1). Note that for each year, we plot only a subset of the submitted systems, which usually does not contain the worst system (except for TectoMT in 2008–2009 and RBMT in 2015).

[2] For example, inconsistencies in the encoding of quotation marks (Unicode typographic vs. plain ASCII) may result in almost one BLEU point difference [Popel and Žabokrtský, 2009].

[3] Different systems may have various sensitivity to the test set “difficulty”.

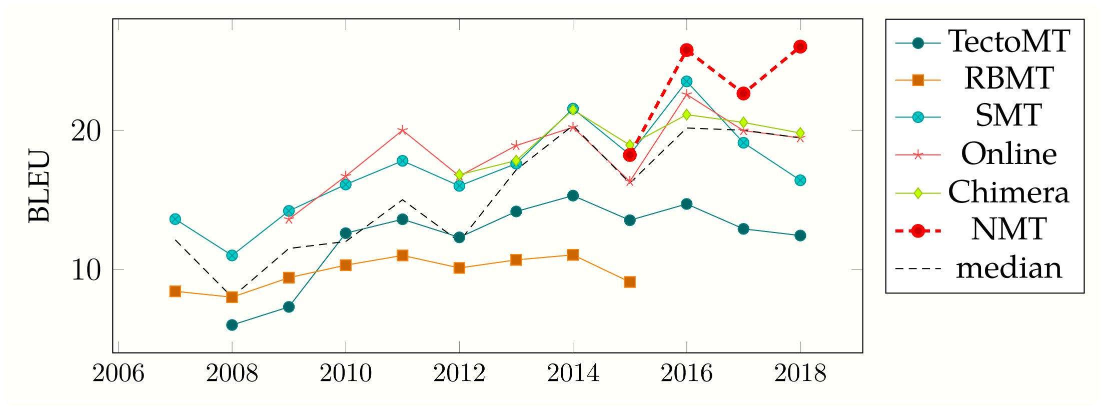


Figure 6.1: WMT English→Czech BLEU evaluation.

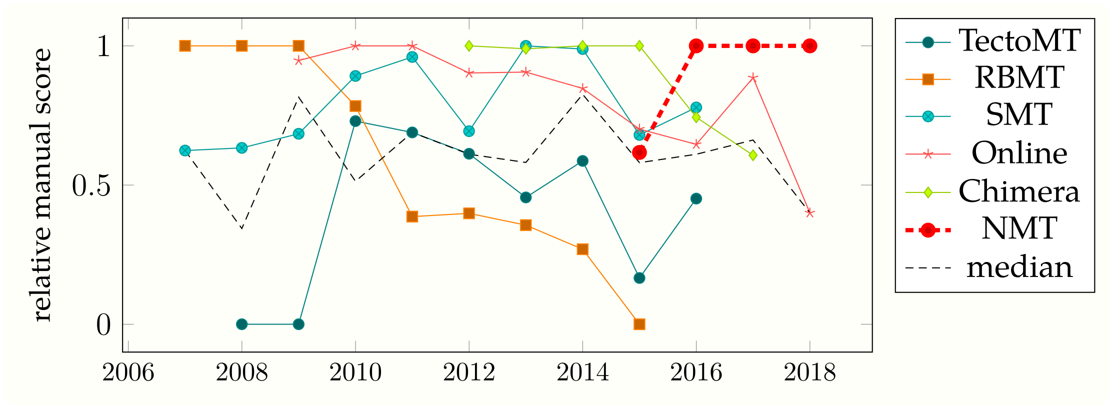


Figure 6.2: WMT English→Czech manual evaluation (higher=better).

year	eval type	TectoMT	RBMT	SMT	Online	Chimera	NMT
2007	≥ranking		**49.9**	*40.5			
2008	≥ranking	49.4	**71.5**	*63.4			
2009	≥ranking	48.0	**67.0**	*61.0	66.0		
2010	≥ranking	60.0	**62.0**	**66.0**	**70.0**		
2011	≥ranking	*57.7	51.1	63.7	**64.6**		
2012	>ranking	*53.1	45.7	*55.9	63.0	**66.4**	
2013	>ranking	47.6	45.7	**58.0**	56.2	**57.8**	
2014	TrueSkill	−17.5	−59.4	**35.6**	16.9	**37.1**	
2015	TrueSkill	20.9	11.4	*50.3	51.5	**68.6**	46.7
2016	TrueSkill	−3.0		34.0	19.0	30.0	***59.0**
2017	srcDA				59.7	54.1	***62.0**
2018	refDA				49.7		***64.1**

Table 6.2: WMT English→Czech manual evaluation scores. The overall winner in a given year is marked with bold (winner means that no other system is statistically significantly better in pairwise comparison; thus there may be more winners each year). The *constrained* winner is marked with * (meaning that no other constrained system is significantly better).

We can conclude with the following observations:

- The best system in 2007–2009 was RBMT, although it had lower BLEU than SMT.[4]
- In 2010–2011, the best system was Online-B, followed by SMT.[5] In 2012, the best system was Depfix post-editing of Online-B (shown here as a Chimera-category system; cf. Section 2.3).
- In 2013–2015, the best system was Chimera: in 2013–2014 it was tied with SMT as the WMT winner, in 2015 it was the sole winner, significantly better than all other systems. All these Chimera systems (2013–2015) included TectoMT (as described in Chapter 3), which contributed essentially to the success of Chimera [Tamchyna and Bojar, 2015]; cf. Section 2.3.
- Since 2016, NMT has been significantly better than all other systems.
- TectoMT was never the overall WMT winner, but in 2011–2012 it was one of the *constrained winners*, i.e. no other constrained system (using only the allowed training data) was significantly better.

Discussion on Paradigm Shifts

We can see paradigm shifts from Rule-based MT to Statistical MT and finally in the last few years to Neural MT (NMT). We focus on English→Czech because for other translation directions, only some of the MT categories are available: TectoMT was submitted to Czech→English WMT only in 2015–2016; Chimera and PC-Translator (RBMT system) were submitted only to English→Czech. However, similar trends in paradigm shifts can be seen in other language pairs as well.

[4] A possible explanation of the mismatch between BLEU and manual ranking is that BLEU as an n-gram based metric is biased towards phrase-based systems [Bojar et al., 2010]. Also, low BLEU scores (say, below 20) have a low correlation with humans in general [Bojar et al., 2010].

[5] Online-B in 2010–2011 is most likely also based on phrase-based SMT.

6.2 WMT2018 Evaluation

In this section, we present results of the automatic and manual evaluation of all systems submitted to WMT2018 for English→Czech and Czech→English

Description of Systems

The set of submitted systems is the same for both directions: three popular online systems with anonymized names (`Online-A`, `Online-B` and `Online-G`) and two academic systems: UEdin-NMT and our Transformer.

UEdin-NMT is the newest version of Nematus system, submitted by a team from the University of Edinburgh, who won the WMT2017 shared task for English→Czech and Czech→English [Sennrich et al., 2017]. According to the available information,[6] it uses a similar setup as in 2017 – an ensemble of eight Deep-RNN models (4 left-to-right and 4 right-to-left) trained with back-translated data. However, the details on training and improvements since last year were not published yet.

Transformer is our system as described in Chapter 5, with the following post-processing (1) and pre-processing (2, 3) adjustments:

1. We deleted phrases repeated more than twice (immediately following each other) from the output; we kept just the first occurrence. We considered phrases of one up to four words.

2. For English→Czech, we converted quotation symbols in the Czech translations to the typographically correct Unicode symbols.

3. For Czech→English, we inserted pronoun *ona* (*she*) into the Czech translations as described below. Our Transformer system translates each sentence independently of other sentences. If the gender information is missing in a sentence, it prefers masculine-geneder translations, e.g. "*Není doma*" is translated as "*He is not home*", although in some context "*She is not home*" is the correct translation.

   We analyzed the documents in Treex (see Section 2.4.1) and found sentences where the female pronoun in subject position was dropped, and the coreference link was pointing to a noun representing a human (i.e. excluding grammatical-only female gender of inanimate objects and animals) in a different sentence. This preprocessing affected only 1% of sentences in our dev set and for most of them the English translation was improved, although the overall BLEU score remained the same. We consider this solution as a temporary workaround before document-level NMT [e.g. Kuang et al., 2017] is available in T2T.

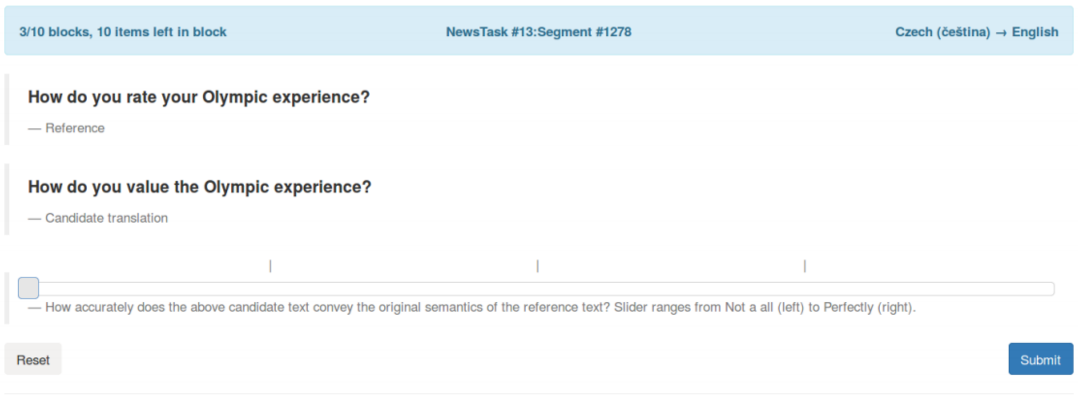
[6] http://matrix.statmt.org/systems/show/3488

Evaluation Description

In the automatic evaluation, we report a character-based chrF2 score and two variants of BLEU score (cf. Section 5.4).

In the manual evaluation results, we report results kindly provided by the WMT organizers (before official publication). The manual annotations were carried out by two groups: annotators hired via the Amazon's Mechanical Turk crowd-sourcing platform (crowd) and researchers (or people hired by the researchers) who submitted the systems.

As mentioned in Section 2.2.1, two types of manual evaluation were carried out in WMT2018: refDA and srcDA. In refDA (reference-based direct assessments) evaluation, the annotators see the system translation of a single sentence together with a reference translation. The task is to assess the quality of the translation relative to the reference on a 0–100 scale. The srcDA (source-based DA) evaluation is similar, but instead of the reference translation, the source sentence is shown to the annotators. In this case, the reference can be used as one of the compared systems, in order to compare the quality of MT relative to the quality of humans. Figure 6.3 shows an example of the annotation interface.



3/10 blocks, 10 items left in block

NewsTask #13:Segment #1278

Czech (čeština) → English

How do you rate your Olympic experience?

— Reference

How do you value the Olympic experience?

— Candidate translation

— How accurately does the above candidate text convey the original semantics of the reference text? Slider ranges from Not a all (left) to Perfectly (right).

Figure 6.3: A screen shot of the Appraise [Federmann, 2012] manual evaluation interface for Czech→English refDA.

Results

Tables 6.3 and 6.4 show the automatic and manual evaluation results for English →Czech. Similarly, Tables 6.5 and 6.6 show the Czech→English. We can see our Transformer system is the best one in both directions in all metrics. Interestingly and surprisingly, it is also significantly better then the human reference translation.

English→Czech

system	BLEU uncased	BLEU cased	chrF2 cased
our Transformer	**26.82**	**26.01**	**0.5372**
UEdin NMT	24.30	23.42	0.5166
Chimera	21.43	19.81	0.4838
Online-B	20.16	19.45	0.4854
Moses	17.88	16.36	0.4594
Online-A	16.84	15.74	0.4584
Online-G	16.33	15.11	0.4560
TectoMT	13.09	12.43	0.4332

Table 6.3: WMT2018 English→Czech automatic evaluation. For reference, we have included also two systems not participating in WMT2018: Moses and TectoMT.

	crowd		researchers			
	srcDA		srcDA		refDA	
system	Avg %	Avg z	Avg %	Avg z	Avg %	Avg z
our Transformer	**79.5**	**0.705**	**84.5**	**0.669**	**64.1**	**0.610**
UEdin NMT	74.1	0.486	79.8	0.521	58.0	0.396
reference	73.1	0.484	78.6	0.483		
Online-B	65.2	0.126	68.1	0.127	49.7	0.113
Online-A	55.6	−0.248	59.4	−0.179	43.9	−0.090
Online-G	50.4	−0.446	54.1	−0.354	40.1	−0.218

Table 6.4: WMT2018 English→Czech manual evaluation. Significantly different scores ($p < 0.05$, Wilcoxon signed-rank test) are separated by horizontal lines.

Czech→English

system	BLEU uncased	BLEU cased	chrF2 cased
our Transformer	**35.64**	**33.91**	**0.5876**
Online-B	34.12	33.06	0.5801
UEdin NMT	33.58	31.78	0.5736
Online-A	28.47	26.78	0.5447
Online-G	25.20	22.53	0.5310
TectoMT	11.36	10.02	0.4501

Table 6.5: WMT2018 Czech→English automatic evaluation.

	crowd		researchers	
	srcDA		refDA	
system	Avg %	Avg z	Avg %	Avg z
our Transformer	**80.6**	**0.546**	**66.5**	**0.462**
UEdin NMT	75.5	0.374	61.3	0.265
reference	72.5	0.284		
Online-B	71.0	0.215	59.5	0.208
Online-A	64.4	−0.058	55.1	0.030
Online-G	59.6	−0.227	50.7	−0.114

Table 6.6: WMT2018 Czech→English manual evaluation. Significantly different scores ($p < 0.05$, Wilcoxon signed-rank test) are separated by horizontal lines.

6.3 Discussion

Hajič et al. [2004] wrote: *"The upper bound for the performance of our system would be translation by humans."* Twelve years afterwards, the authors of Google Neural Machine Translation system (GNMT) reported in their article "[GNMT]: Bridging the Gap between Human and Machine Translation" [Wu et al., 2016] that *"In some cases human and GNMT translations are nearly indistinguishable on the relatively simplistic and isolated sentences sampled from Wikipedia and news articles for this experiment."* However, the human translations by "*average bilingual human translators*" were still evaluated as (slightly) better than GNMT for all six tested language pairs (en↔es, en↔fr, en↔zh).

Less than two years afterwards, the manual srcDA evaluation results for English↔Czech (presented in Tables 6.4 and 6.6) show that Transformed significantly outperformed human translation and they suggest we should reconsider the presupposition of Hajič et al. [2004].

Naturally, the question arises how is it possible that an MT system is *evaluated as* better than humans. In this section, we would like to discuss whether we can omit the two *emphasized* words from the previous sentence. While the final answer is still inconclusive, we believe the discussion below provides insight which may be useful in further improvements of (N)MT systems, as well as improvements of MT-evaluation methods.

We first discuss a concern about the quality of reference translations. Subsequently, we present several hypotheses trying to explain the surprising and difficult-to-believe results. We focus on the English→Czech direction, but some of our hypotheses are relevant also for the opposite direction.

Our discussion is partially based on inspecting translation examples from the `wmt18` test set. We uploaded the test set to `http://wmt.ufal.cz/` and used the online MT-ComparEval service [Klejch et al., 2015] for exploring the translations. MT-ComparEval was designed for comparing two MT-systems relative to the reference, not for comparing the quality of an MT system and the reference. Nevertheless, its visualization capabilities proved useful even for the latter purpose.

Concerns about non-professional references and non-calibrated evaluation

Our first concern was that the quality of WMT2018 human references is lower than in other years, perhaps produced by unskilled translators. However, this is not the case: English-Czech references in WMT2018 were created by a well-known professional translation agency[7] "*with a layer of quality control, and a small quality control check by [the WMT organizers] on a random sample*".[8]

While we spot some (usually minor) translation errors in the references (see below), we have no reason to suspect the translations were not created by professional translators, most likely of a higher quality than *average bilingual human translators*.

Our second concern were "*non-calibrated raters and translators with a varying level of proficiency*" mentioned by Wu et al. [2016] as a possible flaw in their evaluation methodology. We are not aware of any such flaws in the WMT srcDA evaluation. Unlike crowd-sourced translations, professional translation agencies should guarantee a stable level of quality and consistency. When reporting srcDA scores (in Tables 6.4 and 6.6), we provided both "Avg %" and "Avg z" measures (see Section 2.2.1). In "Avg z", all scores are *standardized* first, i.e. *calibrated*, so that the scores from each annotator have zero mean and standard deviation equal to one. In addition, srcDA evaluation methodology includes several other techniques that improve consistency and reliability; see [Graham et al., 2016] for details.

Even professional human references contain errors

Despite originating from professional translators, several errors are observed in the WMT reference translations each year.

For example, there is a grammatical error in subject-predicate agreement in the reference translation in the following sample. Verb form *měli* (they had) is not correct in this sentence because the referenced subject *myši* (mice) has a feminine gender in Czech; the correct form is *měly*.

src-orig	The more tryptophan the mice had in their diet, the more of these immune cells they had.'
ref	Čím více tryptofanu měly myši ve **své** stravě, tím více **měli** těchto imunitních buněk."
our	Čím více tryptofanu měly myši v potravě, tím více těchto imunitních buněk **měly**."

Reference translations are affected by translationese effects

In the previous example, we can also see that the reference translation translates *in their diet* as *ve své stravě*. Although this is grammatically correct, using the possessive pronoun in this context is not natural in Czech – the relation between

[7] Translated.net (`https://www.translated.net/en/`)

[8] Personal communication with the WMT organizers.

diet and *mice* is implicit in the Czech sentence even when omitting the pronoun. Using extra possessive pronouns is a common (negative) translationese effect, when translating from English, where the possessive pronoun is necessary in this context.

Another translationese effect can be seen in the following sentence. The human translator decided to translate *including* as *Patří mezi ně*, so that the two names can be translated in nominative case with no suffix changes from the English original. This may be the preferred translation (and thus a positive translationese effect) in academic or legal texts, where the exact spelling of names is important. However, this sentence originates from a story in the Time Magazine, so the more natural-sounding Transformer translation (with names in genitive) seems appropriate (as well).

src-orig	Some of the top OPM leaders hail from Paniai, including Tadius Yogi and Daniel Yudas Kogoya.
ref	Někteří z vůdců OPM pocházejí z Paniai. Patří mezi **ne** Tadius Yogi a Daniel Yudas Kogoya.
our	Někteří z nejvyšších představitelů OPM pocházejí z Paniai, včetně **Tadiuse Yogiho** a **Daniela Yudase Kogoyi**.

Humans are prone to typos

The reference translation in the previous example contains a typographical error: *ne* (no) instead of *ně* (them). As both words exist in Czech, this error cannot be reported by simple (context-unaware) spell-checkers.

The annotators do not see any context

In the srcDA type of evaluation the annotators see always just a single sentence and its translation. They are restricted in the same way as most current MT systems, which also translate each sentence independently. In the following example, the reference uses a masculine form of the verb, but Transformer feminine. While Dixie is a common female first name, in this example it is actually a male surname, so the reference is correct. However, annotators did not have this information, so they could assign better score to the incorrect Transformer translation.

src	Dixie claimed that Sally and her ex-boyfriend Lewis Sprotson were fighting on the fateful night.
ref-orig	Dixie **tvrdil**, že Sally a její ex přítel Lewis Sprotson se osudné noci hádali.
our	Dixie **tvrdila**, že se Sally a její expřítel Lewis Sprotson v osudnou noc hádali.

Note that this phenomenon does not imply that the Transformer translation will be necessarily scored better than the reference in human evaluation. This is because in both srcDA and refDA, the annotators see always just a single translation and assign it an absolute-quality score; they do not rank alternative translations. The sentences are presented to the annotators in random order. Even if an alternative translation of the same source sentence is presented to the same annotator, it may be several days after scoring the first translation and the annotators are highly unlikely to remember which score (0–100) they had assigned. Note that this property is not considered as a flaw of DA, by the authors of DA [Graham et al., 2016].

The original is unfaithful to the translation

Note that half of the 3000 sentences in `wmt18` was originally written in Czech and the other half in English. Already in the previous example, the "reference" was actually not a translation but an original Czech sentence (from a Czech tabloid newspaper `blesk.cz`).[9]

Similarly, in the following example, the reference originates from a Czech newspaper and uses *historical present* (or rather *narrative present*), i.e. the verb *pokračuje* (continues) is in present tense.

src	"Central Bohemian region was most affected," Zaoralová **continued**.
ref-orig	"Nejvíce zasahovali ve Středočeském kraji," **pokračuje** Zaoralová.
our	„Nejvíce to postihlo Středočeský kraj," **pokračovala** Zaoralová.

While in English, *historical present* '*is also used in fiction, for "hot news" (as in headlines)*' [Huddleston and Pullum, 2002, p. 129], the translator considered it more appropriate to use simple past tense (*continued*) in the English translation (which is part of the body of the article, not the headline). However, the annotators (probably) did not know that the source sentence is actually a translation and they (surely) did not see the whole article. Thus, it is very likely they assigned a better score to the Transformer translation, which is more faithful to the source sentence than the original (ref-orig). Transformer is faithful to the source not only in preserving the past tense, but, more importantly, also in the meaning of the quotation – the ref-orig sentence actually means "[They] intervened in [the] Central-Bohemian region."

As discussed above, in the DA evaluation, the annotators do not see alternative translations side by side and they do not rank their relative quality. However, in this case we believe[10] the "unfaithfulness" of ref-orig to the source sentence is so pronounced that even independent annotators would score ref-orig as absolutely–low-quality translation relative to the source sentence.

[9] The story takes place in Croydon (UK) and the article refers to the Mirror tabloid, so it is possible that some parts of the article are actually translations.

[10] The WMT2018 manual evaluation results have not been released yet (July 2018).

WMT organizers normalize the source and reference sentences

Typographically-inclined readers may have noticed that the ref-orig sentence in the previous example uses straight "ASCII" quotation marks, while Transformer uses correct-Czech „lower and upper“ quotes. In English, the distinction between "straight" and “curly” quotes is considered as a rather typographical (or style-related) issue.[11] However, in Czech, a mismatch between lower (opening) and upper (closing) quotes is considered as an error in formal writing.

Interestingly, after inspecting the original web-version of the article,[12] we observed that it uses correct Czech quotes. It seems the sentences (used as the source for the Czech→English evaluation and as the reference for the English→Czech evaluation, presented in this discussion) were *normalized* by the WMT organizers. While such normalization may be beneficial in training data preprocessing (or rather used to be beneficial in the SMT-quality era), we consider it inappropriate for manual MT evaluation in the NMT-quality era. It may bias the comparison of human references with typography-aware MT.[13]

Document-level context is not shown

One of the design decisions of the DA evaluation (both srcDA and refDA) is that the annotators are presented with only a single translation per screen and that the sentences are shuffled randomly. Thus, the annotators cannot see the context of previous and following sentences. If they could see that in the last example, previous sentences mention *firefighters attending to incidents in Central Bohemia*, the scores assigned to the ref-orig sentence could have been higher.

Limited-context evaluation is problematic also for various other cross-sentence phenomena, most notably coreference. For example, “*It is long.*” can be translated as “*Je dlouhá.*” or “*Je dlouhý.*”, in dependence on whether the referred entity is of feminine or masculine gender. This particular example is more problematic in the refDA evaluation, where the annotators naturally expect that the reference is correct with regards to discourse. However, if the previous sentence was “*I like that part.*”, *long* can be still translated in both ways (*dlouhý* and *dlouhá*), in dependence on whether *part* is translated with a feminine noun *část* or with a masculine noun *úsek*.

The described problems could be alleviated if a longer source-side context (a whole document) was shown to the annotators, at the cost of a slower annotation.

We consider this last hypothesis about limited evaluation context one of the most important ones. To summarize:

[11] Note that this thesis is typeset in TeX Gyre Pagella font (based on Palatino), which uses a different shape of quotes than e.g. Times New Roman; see `https://tex.stackexchange.com/a/12103`.

[12] `http://www.blesk.cz/clanek/zpravy-udalosti/485329/`

[13] Of course, some annotators may not consider the distinction in quotes important. Analyzing the manual evaluation results in this aspect is a future work.

- The source-reference pair is translated by humans *within the context of the whole document* (in one direction in case of src-orig and the other direction in case of ref-orig).
- The source-Transformer pair is translated *with no cross-sentence context* (except for the "ona" adjustment mentioned in Section 6.2).
- The manual evaluation is done *with no cross-sentence context*.

We found many other examples where the Transformer translation seemed to represent better the meaning of the source relative to the reference and the reason was the above-mentioned mismatch.

Chapter 7

Conclusions

Translation is the art of failure.
Umberto Eco

In this thesis, we explored and improved two distinct MT systems based on different paradigms: deep-syntactic TectoMT and neural-MT (NMT) Transformer. Our NMT system produces substantially higher-quality translations than TectoMT, in accordance with the recent paradigm shift in the MT field, where NMT is the new state of the art already since 2016. Nevertheless, we hope some of our achievements in the deep-syntactic MT research may be useful in future, even if the TectoMT system itself has little practical use today. Below, we summarize the main achievements of this thesis, divided into two parts related to our two systems.

Achievements in Deep-Syntactic MT

- We designed and implemented a novel context-sensitive discriminative translation model. We applied it to the transfer of lemmas and formemes on the tectogrammatical layer in the TectoMT system. The model uses a number of contextual features extracted from the source-language dependency trees. We compared two implementations of the model: one based on MaxEnt and the other on the VowpalWabbit machine-learning toolkit.
  - With MaxEnt (a maximum-entropy batch optimizer), we built a separate classifier for each source-language lemma. We improved the English→Czech translation quality by +1.4 BLEU compared to a baseline relative-frequency-based model.
  - With VowpalWabbit (an advanced online-learning optimizer), we built a single classifier for all lemmas. We proposed novel *label-dependent* features shared across multiple source lemmas. We improved the translation quality by further +0.2 BLEU relative to MaxEnt. Moreover, training with VowpalWabbit is more than 1000 times faster relative to MaxEnt.
- We adapted TectoMT to new language pairs and domains (IT-domain helpdesk questions and answers) within an international research project QTLeap. We exploited the modular design of TectoMT and analysis-transfer-synthesis design, where the English analysis is shared for all translation directions from English (English→Czech, English→Spanish,

English→Portuguese, English→Dutch and English→Basque), and similarly, the English synthesis is shared for all translations into English. Using the VowpalWabbit translation model for domain adaptation improved the results by +1.6 BLEU compared with a MaxEnt-based domain adaptation. The improved TectoMT system outperformed the SMT system Moses in the IT-domain evaluation in five out of ten language pairs.

- In our opinion, the main advantages of the TectoMT approach are the following: Its sentence representation is linguistically interpretable and its translation strengths are complementary to SMT, as shown by their successful combination in Chimera, the best-performing English→Czech MT system in WMT 2013–2015. The main disadvantages of TectoMT are: It is restricted to 1-best analysis in its main components: tagging, parsing, transfer and synthesis. It is a complex system from the software point of view and its development requires a combination of machine-learning, programming and linguistic skills.

Achievements in Neural MT

- Transformer model is the current state of the art in sequence-to-sequence modelling. We used it for English-Czech NMT and optimized its training. We analyzed in detail the effect of several training hyper-parameters. For example, we observed that larger batch sizes lead not only to faster training, but also to better translation quality. We observed surprising benefits of multi-GPU training: training a model until achieving a given translation quality (as measured by BLEU) is more than three times faster on two GPUs relative to a single GPU.

- We proposed a novel technique for the exploitation of monolingual training data. We observed a surprising synergy (+0.9 BLEU) when using our novel *concat-regime backtranslation* together with *checkpoint averaging*. We gained further improvements (+0.8 BLEU) by iterating the process of *concat-regime backtranslation*.

- We explored the translation differences when training and/or testing on Czech-origin and non-Czech-origin texts. By tuning two separate models for these two tasks, we obtained a further +0.2 BLEU improvement.

Achieved Goals and Future Work

We believe our work achieved all the three goals listed in Section 1.2. The achievements open new research questions and point to interesting future work:

- We improved the English↔Czech MT. Our neural system was evaluated as significantly ($p < 0.05$) better than all other MT systems and also than the human reference translation within the WMT2018 shared task. Training the neural system for other language pairs is straightforward and we plan to do so in the near future. A demo of our English↔Czech NMT is available online, as well as all WMT2018 translation outputs.[1]

- We explored the impact of syntactic structures within our linguistically-motivated TectoMT system. We provided insight into the deep-syntactic translation process, e.g. by interpreting how the context-sensitive model decides which translation to choose and which linguistic features are important for this decision.

  The Transformer model uses *latent* syntactic structures learned in an unsupervised way. However, it remains a future work to explore how exactly are these structures similar to or different from the linguistic structures in manually-annotated treebanks, and thus possibly better understand how NMT (and human language) works.

- We explored domain-adaptation techniques. We successfully adapted TectoMT to a new domain of translating IT helpdesk questions and answers. We improved the translation quality (according to BLEU) of our NMT system by adapting it to the translation of Czech-origin and non-Czech-origin news articles. We discussed relations to the effect of *translationese*, i.e which language in parallel training data is the original and which is the translation.

  We did not perform a manual evaluation of this Czech-origin domain adaptation. Taking into account that the quality of our NMT is higher than the quality of reference translations, we cannot be sure whether the +0.2 BLEU difference is actually an improvement, or whether our translations only resemble more the human references. It is not clear whether the goal of MT is to simulate human translators (and produce "translationese" output) or whether it should rather produce output that resembles texts written originally in the target language.

  Finally, there is a challenging future work on domain adaptation to more difficult domains and genres, e.g. legal texts or fiction.

[1] `https://lindat.mff.cuni.cz/services/transformer/`
`http://wmt.ufal.cz`

Bibliography

Eneko Agirre, Arantxa Otegi, Gorka Labaka, Chakaveh Saedi, João Rodrigues, João Silva, Martin Popel, Roman Sudarikov, Kiril Simov, and Petya Osenova. Report on mt improved with semantic linking and resolving. Technical Report Deliverable D5.11, QTLeap Project, 2016. URL `http://qtleap.eu/wp-content/uploads/2016/11/QTLEAP-2016-D5.11.pdf`.

Nora Aranberri, Eleftherios Avramidis, Aljoscha Burchardt, Ondřej Klejch, Martin Popel, and Maja Popović. Tools and guidelines for principled machine translation development. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1877–1882, Paris, France, 2016. European Language Resources Association. ISBN 978-2-9517408-9-1.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised Neural Machine Translation. In *International Conference on Learning Representations*, volume arXiv/1710.11041, 2018. URL `http://arxiv.org/abs/1710.11041`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, arXiv/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Mona Baker. *Routledge Encyclopedia of Translation Studies*. Routledge, London, 1998. ISBN 0-415-09380-5.

Mona Baker, Gill Francis, and Elena Tognini-Bonelli. *Corpus Linguistics and Translation Studies: Implications and Applications*. John Benjamins Publishing Company, Netherlands, 1993.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. Evaluating Discourse Phenomena in Neural Machine Translation. *CoRR*, arXiv/1711.00513, 2017. URL `http://arxiv.org/abs/1711.00513`.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1): 39–71, 1996.

Nicola Bertoldi and Marcello Federico. Domain Adaptation for Statistical Machine Translation with Monolingual Resources. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 182–189, Athens, Greece, March 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W09-0432`.

Ondřej Bojar and Jan Hajič. Phrase-Based and Deep Syntactic English-to-Czech Statistical Machine Translation. In *ACL 2008 WMT: Proceedings of the Third Workshop on Statistical Machine Translation*, pages 143–146, Columbus, OH, USA, 2008. Association for Computational Linguistics. ISBN 978-1-932432-09-1.

Ondřej Bojar and Zdeněk Žabokrtský. Building a Large Czech-English Automatic Parallel Treebank. *Prague Bulletin of Mathematical Linguistics*, 92, 2009.

Ondřej Bojar, Kamil Kos, and David Mareček. Tackling Sparse Data Issue in Machine Translation Evaluation. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 86–91, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P10-2016`.

Ondřej Bojar, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. The Joy of Parallelism with CzEng 1.0. In *Proceedings of the Eighth International Language Resources and Evaluation Conference (LREC'12)*, pages 3921–3928, Istanbul, Turkey, May 2012. ELRA, European Language Resources Association. ISBN 978-2-9517408-7-7.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, number 9924 in Lecture Notes in Artificial Intelligence, pages 231–238. Masaryk University, Springer International Publishing, 2016a. ISBN 978-3-319-45509-9.

Ondrej Bojar, Christian Federmann, Barry Haddow, Philipp Koehn, Matt Post, and Lucia Specia. Ten Years of WMT Evaluation Campaigns: Lessons Learnt. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016b. URL `http://www.cs.jhu.edu/~post/papers/bojar2016ten.pdf`.

Ondřej Bojar. English-to-Czech factored machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 232–239, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-0735`.

Ondřej Bojar. *Exploiting Linguistic Data in Machine Translation*. PhD thesis, ÚFAL, MFF UK, Prague, Czech Republic, October 2008.

Ondřej Bojar and Aleš Tamchyna. Improving Translation Model by Monolingual Data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 330–336, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W11-2138`.

Ondřej Bojar and Aleš Tamchyna. CUNI in WMT15: Chimera Strikes Again. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 79–83, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL `http://aclweb.org/anthology/W15-3006`.

Ondřej Bojar, Miloš Ercegovčević, Martin Popel, and Omar Zaidan. A Grain of Salt for the WMT Manual Evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 1–11, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2101.

Ondřej Bojar, Rudolf Rosa, and Aleš Tamchyna. Chimera – Three Heads for English-to-Czech Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 92–98, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2208.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016c. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2301.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark, September 2017a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W17-4717.

Ondřej Bojar, Yvette Graham, and Amir Kamran. Results of the WMT17 Metrics Shared Task. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 489–513, Copenhagen, Denmark, September 2017b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W17-4755.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *CoRR*, arXiv/1606.04838, June 2016. URL https://arxiv.org/abs/1606.04838.

Léon Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_25. URL https://doi.org/10.1007/978-3-642-35289-8_25.

Peter F Brown, Robert L Mercer, TJ Watson, Vincent J Della Pietra, Jenifer C Lai, et al. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18(4), 1992. URL http://www.aclweb.org/anthology/J92-4003.

Peter F. Brown, Stephen A. Della-Pietra, Vincent J. Della-Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation. *Computational Linguistics*, 19(2): 263–313, 1993. URL http://www.aclweb.org/anthology/J/J93/J93-2003.pdf.

Aljoscha Burchardt and Eleftherios Avramidis. Report on evaluation metrics and baselines for the project. Technical report, QTLeap Project, 2015. URL http://qtleap.eu/wp-content/uploads/2015/06/QTLEAP-2015-D2.2.pdf.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluation the Role of Bleu in Machine Translation Research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006. URL `http://www.aclweb.org/anthology/E06-1032`.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. (Meta-) Evaluation of Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-0218`.

Jaime G. Carbonell, Teruko Mitamura, and Eric Nyberg. The KANT Perspective: A Critique of Pure Transfer (and Pure Interlingua, Pure Statistics,...). In *Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 225–235, Montreal, Canada, 1992. doi: 10.1184/R1/6610655.v1. URL `http://mt-archive.info/TMI-1992-Carbonell.pdf`.

M. Asunción Castaño, Francisco Casacuberta, and Enrique Vidal. Machine translation using neural networks and finite-state models. In *"Theoretical and Methodological Issues in Machine Translation"*, pages 160–167, 1997. URL `http://www.mt-archive.info/TMI-1997-Castano.pdf`.

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of Association for Computational Linguistics*, ACL '05, pages 173–180, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL `http://dx.doi.org/10.3115/1219840.1219862`.

David Chiang. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P05/P05-1033`.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D14-1179`.

Lonnie Chrisman. Learning recursive distributed representations for holistic computation. *Connection Science*, 3(4):345–366, 1991. URL `http://www.chrisman.org/Lonnie/papers/chrisman_cmu_cs_91_154.pdf`.

Martin Čmejrek. *Using Dependency Tree Structure for Czech-English Machine Translation*. PhD thesis, ÚFAL, MFF UK, Prague, Czech Republic, 2006.

Martin Čmejrek, Jan Cuřín, and Jiří Havelka. Czech-english dependency-based machine translation. In Ann Copestake and Jan Hajič, editors, *EACL 2003 Proceedings of the Conference*, pages 83–90, Budapest, Hungary, 2003. Association for Computational Linguistics. ISBN 1-932432-00-0. URL `https://doi.org/10.3115/1067807.1067820`.

Jan Cuřín, Martin Čmejrek, Jiří Havelka, Jan Hajič, Vladislav Kuboň, and Zdeněk Žabokrtský. Prague Czech-English Dependency Treebank, Version 1.0. Linguistics Data Consortium, Catalog No.: LDC2004T25, 2004.

Michael Denkowski and Graham Neubig. Stronger Baselines for Trustable Results in Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27. Association for Computational Linguistics, 2017. URL http://aclweb.org/anthology/W17-3203.

Ondřej Dušek, Eva Fučíková, Jan Hajič, Martin Popel, Jana Šindlerová, and Zdeňka Urešová. Using Parallel Texts and Lexicons for Verbal Word Sense Disambiguation. In Eva Hajičová and Joakim Nivre, editors, *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 82–90, Uppsala, Sweden, 2015a. Uppsala University, Uppsala University. ISBN 978-91-637-8965-6.

Ondřej Dušek, Luís Gomes, Michal Novák, Martin Popel, and Rudolf Rosa. New Language Pairs in TectoMT. In *Proceedings of the 10th Workshop on Machine Translation*, pages 98–104, Stroudsburg, PA, USA, 2015b. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-941643-32-7. URL http://www.emnlp2015.org/proceedings/WMT/pdf/WMT09.pdf.

Ondřej Dušek, Zdeněk Žabokrtský, Martin Popel, Martin Majliš, Michal Novák, and David Mareček. Formemes in English-Czech Deep Syntactic MT. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 267–274, Montréal, Canada, 2012. Association for Computational Linguistics.

Ondřej Dušek, Martin Popel, Vivien Macketanz, Aljoscha Burchardt, Eleftherios Avramidis, Gertjan van Noord, João Rodrigues, António Branco, Gorka Labaka, Kiril Simov, and Aleksandar Popov. Report on the third mt pilot and its evaluation. Technical Report Deliverable D2.11, QTLeap Project, 2016. URL http://qtleap.eu/wp-content/uploads/2016/11/QTLEAP-2016-D2.11.pdf.

Jason Eisner. Learning Non-Isomorphic Tree Mappings for Machine Translation. In *Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 205–208, Sapporo, Japan, July 2003. Association for Computational Linguistics. doi: 10.3115/1075178.1075217. URL http://www.aclweb.org/anthology/P03-2039.

R. E Fan, K. W Chang, C. J Hsieh, X. R Wang, and C. J Lin. LIBLINEAR: a library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

Christian Federmann. Appraise: An Open-Source Toolkit for Manual Evaluation of Machine Translation Output. *The Prague Bulletin of Mathematical Linguistics*, 98:25–35, September 2012. URL https://ufal.mff.cuni.cz/pbml/98/art-federmann.pdf.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

Mikel L. Forcada and Ramón P. Ñeco. Recursive hetero-associative memories for translation. In *Biological and Artificial Computation: From Neuroscience to Technology*, pages 453–462. Springer, 1997.

Mikel L. Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M. Tyers. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144, Jun 2011. ISSN 1573-0573. doi: 10.1007/s10590-011-9090-0. URL https://doi.org/10.1007/s10590-011-9090-0.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, 2004. URL http://www.aclweb.org/anthology/N04-1035.pdf.

Kuzman Ganchev and Mark Dredze. Small statistical models by random feature mixing. In *Proceedings of the ACL-08: HLT Workshop on Mobile Language Processing*, pages 19–20, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W08/W08-0804.

Rosa Del Gaudio, Aljoscha Burchardt, António Branco, and Martin Popel. Report on the embedding and evaluation of the third mt pilot. Technical report, QTLeap Project, 2016. URL http://qtleap.eu/wp-content/uploads/2016/11/QTLEAP-2016-D3.12.pdf.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional Sequence to Sequence Learning. *CoRR*, arXiv/1705.03122, 2017. URL http://arxiv.org/abs/1705.03122.

Martin Gellerstam. Translationese in Swedish novels translated from English. In Lars Wollin and Hans Lindquist, editors, *Translation Studies in Scandinavia*, pages 88–95. CWK Gleerup, 1986.

Zhengxian Gong, Min Zhang, and Guodong Zhou. Document-level machine translation evaluation with gist consistency and text cohesion. In *Proceedings of the Second Workshop on Discourse in Machine Translation*, pages 33–40, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/W15-2504.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *CoRR*, arXiv/1706.02677, 2017. URL http://arxiv.org/abs/1706.02677.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, FirstView:1–28, 1 2016. ISSN 1469-8110. doi: 10.1017/S1351324915000339. URL http://journals.cambridge.org/article_S1351324915000339.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-Autoregressive Neural Machine Translation. *CoRR*, arxiv/1711.02281, 2017. URL http://arxiv.org/abs/1711.02281.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On Using Monolingual Corpora in Neural Machine Translation. *CoRR*, arXiv/1503.03535, 2015. URL http://arxiv.org/abs/1503.03535.

Jan Hajič. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Nakladatelství Karolinum, 2004. ISBN 80-246-0282-2.

Jan Hajič, Martin Čmejrek, Bonnie Dorr, Yuan Ding, Jason Eisner, Dan Gildea, Terry Koo, Kristen Parton, Gerald Penn, Dragomir Radev, and Owen Rambow. Natural Language Generation in the Context of Machine Translation. Technical report, CLSP JHU, USA, 2004. URL `http://www.cs.jhu.edu/~jason/papers/hajic+al.ws02.pdf`.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, and Marie Mikulová. Prague Dependency Treebank 2.0. CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia, 2006.

Jan Hajič, Silvie Cinková, Kristýna Čermáková, Lucie Mladová, Anja Nedolužko, Pajas Petr, Jiří Semecký, Jana Šindlerová, Josef Toman, Kristýna Tomšů, Matěj Korvas, Magdaléna Rysová, Kateřina Veselovská, and Zdeněk Žabokrtský. Prague English Dependency Treebank, Version 1.0, Jan 2009.

J. Hajič, E. Hajičová, J. Panevová, P. Sgall, O. Bojar, S. Cinková, E. Fučíková, M. Mikulová, P. Pajas, J. Popelka, J. Semecký, J. Šindlerová, J. Štěpánek, J. Toman, Z. Urešová, and Z. Žabokrtský. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of LREC*, pages 3153–3160, Istanbul, 2012.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, Boulder, Colorado, USA, 2009.

Jiří Hana. *Czech Clitics in Higher Order Grammar*. PhD thesis, The Ohio State University, 2007. URL `http://ufal.mff.cuni.cz/~hana/bib/hana-diss.pdf`.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1731–1741. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6770-train-longer-generalize-better-closing-the-generalization-gap-in-large-batch-training-of-neural-networks.pdf.

Liang Huang, Kevin Knight, and Aravind Joshi. A Syntax-Directed Translator with Extended Domain of Locality. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 1–8, New York City, New York, June 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W06/W06-3601`.

Rodney Huddleston and Geoffrey K. Pullum. *The Cambridge Grammar of the English Language*. Cambridge University Press, Cambridge, UK New York, 2002. ISBN 0521431468.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, arXiv/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging Weights Leads to Wider Optima and Better Generalization. *CoRR*, arXiv/1803.05407, 2018. URL http://arxiv.org/abs/1803.05407.

Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos J. Storkey. Three factors influencing minima in SGD. *CoRR*, arXiv/1711.04623, 2017. URL http://arxiv.org/abs/1711.04623.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. Montreal Neural Machine Translation Systems for WMT'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/W15-3014.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. The AMU-UEDIN Submission to the WMT16 News Translation Task: Attention-based NMT Models as Feature Functions in Phrase-based SMT. In *Proceedings of the First Conference on Machine Translation*, pages 319–325, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W16-2316.

Nal Kalchbrenner and Phil Blunsom. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D13-1176.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *Proceedings of ICLR*, volume arXiv/1609.04836, 2017. URL http://arxiv.org/abs/1609.04836.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, arXiv/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Zdeněk Kirschner and Alexandr Rosen. APAČ – An Experiment in Machine Translation. *Machine Translation*, 4(3):177–193, 1989.

Zdeněk Kirschner. Appendix. In *In Automatische Textenbearbeitung*, pages 86–156. Matematicko-fyzikální fakulta Univerzity Karlovy, Prague, 1974. Published anonymously.

Zdeněk Kirschner. A dependency-based analysis of English for the purpose of machine translation. In *Explizite Beschreibung der Sprache und automatische Textbearbeitung*. Matematicko-fyzikální fakulta Univerzity Karlovy, Prague, 1982.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of Association for Computational Linguistics*, pages 423–430, 2003.

Ondřej Klejch, Eleftherios Avramidis, Aljoscha Burchardt, and Martin Popel. MT-ComparEval: Graphical evaluation interface for Machine Translation development. *The Prague Bulletin of Mathematical Linguistics*, 104:63–74, October 2015. ISSN 0032-6585. doi: 10.1515/pralin-2015-0014. URL `http://ufal.mff.cuni.cz/pbml/104/art-klejch-et-al.pdf`.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, volume 4, pages 388–395, 2004.

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2010.

Philipp Koehn and Hieu Hoang. Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, 2007. URL `http://www.aclweb.org/anthology/D07-1091`.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Christopher J. Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P07/P07-2045`.

Moshe Koppel and Noam Ordan. Translationese and Its Dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1318–1326, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1132`.

Kamil Kos and Ondřej Bojar. Evaluation of Machine Translation Metrics for Czech as the Target Language. *Prague Bulletin of Mathematical Linguistics*, 92, 2009. ISSN 0032-6585.

Jana Kravalová. Využití syntaxe v metodách pro vyhledávání informací (using syntax in information retrieval). Master's thesis, Faculty of Mathematics and Physics, Charles University in Prague, 2009.

Jana Kravalová and Zdeněk Žabokrtský. Czech Named Entity Corpus and SVM-based Recognizer. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 194–201, Suntec, Singapore, 2009. Association for Computational Linguistics. ISBN 978-1-932432-57-2.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, arXiv/1404.5997, 2014. URL `http://arxiv.org/abs/1404.5997`.

Shaohui Kuang, Deyi Xiong, Weihua Luo, and Guodong Zhou. Cache-based Document-level Neural Machine Translation. *CoRR*, arXiv/1711.11221, 2017. URL `http://arxiv.org/abs/1711.11221`.

Taku Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. *CoRR*, arXiv/1804.10959, 2018. URL `http://arxiv.org/abs/1804.10959`.

John Langford, Lihong Li, and Alex Strehl. Vowpal Wabbit online learning project, 2007. Technical report, `http://hunch.net/~vw`.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully Character-Level Neural Machine Translation without Explicit Segmentation. *CoRR*, arXiv/1610.03017, 2016. URL `http://arxiv.org/abs/1610.03017`.

J. Lei Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *CoRR*, arXiv/1607.06450, July 2016.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, Aug 1989. ISSN 1436-4646. doi: 10.1007/BF01589116. URL `https://doi.org/10.1007/BF01589116`.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Restarts. *CoRR*, arXiv/1608.03983, 2016. URL `http://arxiv.org/abs/1608.03983`.

Minh-Thang Luong and Christopher Manning. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 76–79, December 2015. URL `http://workshop2015.iwslt.org/downloads/IWSLT_2015_EP_19.pdf`.

Minh-Thang Luong and Christopher D. Manning. Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany, August 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P16-1100`.

D. Macháček, J. Vidra, and O. Bojar. Morphological and Language-Agnostic Word Segmentation for NMT. *CoRR*, June 2018. URL `http://arxiv.org/abs/1806.05482`.

Matouš Macháček and Ondřej Bojar. Results of the WMT13 Metrics Shared Task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 45–51, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-2202`.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2): 313–330, 1994.

David Mareček, Zdeněk Žabokrtský, and Václav Novák. Automatic Alignment of Czech and English Deep Syntactic Dependency Trees. In John Hutchins and Walther Hahn, editors, *Proceedings of the Twelfth EAMT Conference*, pages 102–111, Hamburg, 2008. HITEC e.V. ISBN 978-3-00-025770-4. URL `http://www.mt-archive.info/EAMT-2008-Marecek.pdf`.

David Mareček, Martin Popel, and Zdeněk Žabokrtský. Maximum Entropy Translation Model in Dependency-Based MT Framework. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 201–201, Uppsala, Sweden, 2010. Uppsala Universitet, Association for Computational Linguistics. ISBN 978-1-932432-71-8. URL `http://www.aclweb.org/anthology/W10-1730`.

David Mareček and Natalia Kljueva. Converting Russian Treebank SynTagRus into Praguian PDT Style. In *Proceedings of the RANLP 2009 (International Conference on Recent Advances in Natural Language Processing)*, Borovets, Bulgaria, 2009.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT / EMNLP*, pages 523–530, Vancouver, Canada, 2005.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR*, arXiv/1301.3781, 2013. URL `http://arxiv.org/abs/1301.3781`.

Robert C. Moore and William Lewis. Intelligent Selection of Language Model Training Data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P10-2041`.

Rebecca Nesson, Stuart M. Shieber, and Alexander Rush. Induction of Probabilistic Synchronous Tree-Insertion Grammars for Machine Translation. In *5th Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, Massachusetts, August 2006. URL `http://www.mt-archive.info/AMTA-2006-Nesson.pdf`.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. Reinforcement Learning for Bandit Neural Machine Translation with Simulated Human Feedback. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1464–1474, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D17-1153`.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gulsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

Franz Josef Och. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1075096.1075117. URL `http://dx.doi.org/10.3115/1075096.1075117`.

Franz Josef Och and Hermann Ney. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting of the Association of Computational Linguistics (ACL)*, 2000. URL `http://www.aclweb.org/anthology/P/P00/P00-1056.pdf`.

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

M. Ott, S. Edunov, D. Grangier, and M. Auli. Scaling Neural Machine Translation. *CoRR*, arXive/1806.00187, June 2018. URL `https://arxiv.org/abs/1806.00187`.

Petr Pajas and Jan Štěpánek. Recent advances in a feature-rich framework for treebank annotation. In *The 22nd International Conference on Computational Linguistics - Proceedings of the Conference*, pages 673–680, 2008. URL `http://www.aclweb.org/anthology/C08-1085`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL 2002*, pages 311–318, Philadelphia, Pennsylvania, 2002.

Pavel Pecina, Ondřej Dušek, Lorraine Goeuriot, Jan Hajič, Jaroslava Hlaváčová, Gareth Jones, Liadh Kelly, Johannes Leveling, David Mareček, Michal Novák, Martin Popel, Rudolf Rosa, Aleš Tamchyna, and Zdeňka Urešová. Adaptation of machine translation for multilingual information retrieval in medical domain. *Artificial Intelligence in Medicine*, 61(3):165–185, 2014. ISSN 0933-3657.

A. Poncelas, D. Shterionov, A. Way, G. Maillette de Buy Wenniger, and P. Passban. Investigating Backtranslation in Neural Machine Translation. *CoRR*, arXiv/1804.06189, April 2018. URL http://arxiv.org/abs/1804.06189.

Martin Popel. Ways to Improve the Quality of English-Czech Machine Translation. Master's thesis, Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic, 2009. URL http://ufal.mff.cuni.cz/~popel/papers/master_thesis.pdf.

Martin Popel and Ondřej Bojar. Training Tips for the Transformer Model. *The Prague Bulletin of Mathematical Linguistics*, 110:43–70, April 2018. ISSN 0032-6585. doi: 10.2478/pralin-2018-0002. URL https://ufal.mff.cuni.cz/pbml/110/art-popel-bojar.pdf.

Martin Popel and David Mareček. Perplexity of n-gram and dependency language models. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue. 13th International Conference, TSD 2010, Brno, Czech Republic, September 6-10, 2010. Proceedings*, volume 6231 of *Lecture Notes in Computer Science*, pages 173–180, Berlin / Heidelberg, 2010. Masarykova univerzita, Springer. ISBN 978-3-642-15759-2. doi: https://doi.org/10.1007/978-3-642-15760-8_23. URL http://ufal.mff.cuni.cz/~popel/papers/2010_tsd.pdf.

Martin Popel and Zdeněk Žabokrtský. Improving English-Czech Tectogrammatical MT. *The Prague Bulletin of Mathematical Linguistics*, 92:1–20, 2009. ISSN 0032-6585.

Martin Popel and Zdeněk Žabokrtský. TectoMT: Modular NLP Framework. *Advances in Natural Language Processing*, pages 293–304, 2010. URL http://ufal.mff.cuni.cz/~popel/papers/2010_icetal.pdf.

Martin Popel, David Mareček, Nathan Green, and Zdeněk Žabokrtský. Influence of Parser Choice on Dependency-Based MT. In Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan, editors, *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 433–439, Edinburgh, UK, 2011. University of Edinburgh, Association for Computational Linguistics. ISBN 978-1-937284-12-1. URL http://www.aclweb.org/anthology/W11-2153.

Martin Popel, Jaroslava Hlaváčová, Ondřej Bojar, Ondřej Dušek, António Branco, Luís Gomes, João Rodrigues, Andreia Querido João Silva, Nuno Rendeiro, Marisa Campos, Diana Amaral, Eleftherios Avramidis, Aljoscha Burchardt, Maja Popovic, Arle Lommel, Iliana Simova, Nora Aranberri, Gorka Labaka, Gertjan van Noord, Rosa Del Gaudio, Michal Novák, Rudolf Rosa, Aleš Tamchyna, and Jan Hajič. Report on the first mt pilot and its evaluation. Technical Report Deliverable D2.4, Version 1.8, QTLeap Project, 2015. URL http://qtleap.eu/wp-content/uploads/2015/04/QTLEAP-2015-D2.4.pdf.

Martin Popel, Zdeněk Žabokrtský, and Martin Vojtek. Udapi: Universal API for Universal Dependencies. In *NoDaLiDa 2017 Workshop on Universal Dependencies*, pages 96–101, Göteborg, Sweden, 2017. Göteborgs universitet. ISBN 978-91-7685-501-0.

Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September 2015. ACL. URL http://aclweb.org/anthology/W15-3049.

Matt Post. A Call for Clarity in Reporting BLEU Scores. *CoRR*, arXiv/1804.08771, April 2018. URL http://arxiv.org/abs/1804.08771.

Spencer Rarrick, Chris Quirk, and Will Lewis. MT Detection in Web-Scraped Parallel Corpora. In *Proceedings of MT Summit XIII*. Asia-Pacific Association for Machine Translation, September 2011. URL https://www.microsoft.com/en-us/research/publication/mt-detection-in-web-scraped-parallel-corpora/.

Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *In Proceedings of EMNLP'96*, pages 133–142, 1996.

Jan Romportl. *Zvyšování přirozenosti strojově vytvářené řeči v oblasti suprasegmentálních zvukových jevů*. PhD thesis, Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech Republic, 2008.

Rudolf Rosa, David Mareček, and Ondřej Dušek. DEPFIX: A System for Automatic Correction of Czech MT Outputs. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 362–368, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-3146.

Rudolf Rosa, Jan Mašek, David Mareček, Martin Popel, Daniel Zeman, and Zdeněk Žabokrtský. HamleDT 2.0: Thirty Dependency Treebanks Stanfordized. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, and Joseph Mariani, editors, *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 2334–2341, Reykjavík, Iceland, 2014. European Language Resources Association. ISBN 978-2-9517408-8-4. URL http://www.lrec-conf.org/proceedings/lrec2014/pdf/915_Paper.pdf.

Rudolf Rosa, Ondřej Dušek, Michal Novák, and Martin Popel. Translation model interpolation for domain adaptation in tectoMT. In Jan Hajič and António Branco, editors, *Proceedings of the 1st Deep Machine Translation Workshop*, pages 89–96, Praha, Czechia, 2015. ÚFAL MFF UK, ÚFAL MFF UK. ISBN 978-80-904571-7-1. URL http://www.aclweb.org/anthology/W15-5711.

Rudolf Rosa, Roman Sudarikov, Michal Novák, Martin Popel, and Ondřej Bojar. Dictionary-based domain adaptation of mt systems without retraining. In *Proceedings of the First Conference on Machine Translation*, pages 449–455, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2334.

Alexandr Rosen. *A constraint-based approach to dependency syntax applied to some issues of Czech word order*. PhD thesis, Faculty of Philosophy, Charles University, Prague, 2001. URL http://utkl.ff.cuni.cz/~rosen/public/THESIS/thesis.pdf.

Aurko Roy, Ashish Vaswani, Arvind Neelakantan, and Niki Parmar. Theory and Experiments on Vector Quantized Autoencoders. *CoRR*, arXiv/1805.11063, 2018. URL http://arxiv.org/abs/1805.11063.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. Efficient Elicitation of Annotations for Human Evaluation of Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 1–11, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3301.

Danielle Saunders, Felix Stahlberg, Adrià de Gispert, and Bill Byrne. Multi-representation Ensembles and Delayed SGD Updates Improve Syntax-based NMT. In *56th Annual Meeting of the Association for Computational Linguistics*, 2018. URL https://arxiv.org/abs/1805.00456.

Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *ICASSP*, pages 5149–5152, 2012. URL https://research.google.com/pubs/archive/37842.pdf.

Holger Schwenk. Investigations on Large-Scale Lightly-Supervised Training for Statistical Machine Translation. In *Proceedings of IWSLT*, pages 182–189, 2008. URL https://www.isca-speech.org/archive/iwslt_08/papers/slt8_182.pdf.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, August 2016a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1009.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of ACL 2016*, pages 1715–1725, Berlin, Germany, August 2016b. ACL. URL http://www.aclweb.org/anthology/P16-1162.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Edinburgh Neural Machine Translation Systems for WMT16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany, August 2016c. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W16-2323.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. The University of Edinburgh's Neural MT Systems for WMT17. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 389–399, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W17-4739.

Petr Sgall. *Generativní popis jazyka a česká deklinace*. Academia, Prague, Czech Republic, 1967.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. D. Reidel Publishing Company, Dordrecht, 1986.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-Attention with Relative Position Representations. *CoRR*, arXiv/1803.02155, 2018. URL http://arxiv.org/abs/1803.02155.

N. Shazeer and M. Stern. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. *CoRR*, arXiv/1804.04235, April 2018. URL https://arxiv.org/abs/1804.04235.

Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *Proceedings of Second workshop on Bayesian Deep Learning (NIPS 2017)*, volume arXiv/1710.06451, Long Beach, CA, USA, 2017. URL http://arxiv.org/abs/1710.06451.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. *CoRR*, arXiv/1711.00489, 2017. URL http://arxiv.org/abs/1711.00489.

Drahomíra Spoustová, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň. 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, ACL 2007*, pages 67–74, Praha, 2007.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Jana Straková, Milan Straka, and Jan Hajič. 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*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P14/P14-5003.pdf.

Sara Stymne. The Effect of Translationese on Tuning for Statistical Machine Translation. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 241–246, Gothenburg, Sweden, May 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W17-0230.

Roman Sudarikov, Martin Popel, Ondřej Bojar, Aljoscha Burchardt, and Ondřej Klejch. Using MT-ComparEval. In Ondřej Bojar, Aljoscha Burchardt, Christian Dugast, Marcello Federico, Josef Genabith, Barry Haddow, Jan Hajič, Kim Harris, Philipp Koehn, Matteo Negri, Martin Popel, Georg Rehm, Lucia Specia, Marco Turchi, and Hans Uszkoreit, editors, *Translation Evaluation: From Fragmented Tools and Data Sets to an Integrated Ecosystem*, pages 76–82, Portorož, Slovenia, 2016. LREC. URL http://www.cracking-the-language-barrier.eu/wp-content/uploads/Sudarikov-etal.pdf.

Roman Sudarikov, David Mareček, Tom Kocmi, Dušan Variš, and Ondřej Bojar. CUNI submission in WMT17: Chimera goes neural. In *Proceedings of the Second Conference* *on Machine Translation, Volume 2: Shared Task Papers*, pages 248–256, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W17-4720.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. URL https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Aleš Tamchyna and Ondrej Bojar. What a Transfer-Based System Brings to the Combination with PBMT. In *Proceedings of the Fourth Workshop on Hybrid Approaches to Translation (HyTra)*, pages 11–20, Beijing, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W15-4103.

Aleš Tamchyna, Martin Popel, Rudolf Rosa, and Ondrej Bojar. CUNI in WMT14: Chimera Still Awaits Bellerophon. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 195–200, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3322.

Aleš Tamchyna, Alexander Fraser, Ondřej Bojar, and Marcin Junczys-Dowmunt. 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)*, pages 1704–1714, Berlin, Germany, August 2016a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1161.

Aleš Tamchyna, Roman Sudarikov, Ondřej Bojar, and Alexander Fraser. CUNI-LMU Submissions in WMT2016: Chimera Constrained and Beaten. In *Proceedings of the First Conference on Machine Translation*, pages 385–390, Berlin, Germany, August 2016b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2325.

Peter Toma. Systran as a multilingual machine translation system. In *Proceedings of the Third European Congress on Information Systems and Networks, Overcoming the language barrier*, pages 569–581, 1977. URL http://www.mt-archive.info/CEC-1977-Toma.pdf.

Gideon Toury. *Descriptive translation studies and beyond*. John Benjamins Publishing Company, 1995.

Joachim Utans. Weight Averaging for Neural Networks and Local Resampling Schemes. In *Proceedings of AAAI-96 Workshop on Integrating Multiple Learned Models*, pages 133–138, 06 1996. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.7218&rep=rep1&type=pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Bernard Vauquois. La traduction automatique à Grenoble. Document de linguistique quantitative 24. Dunod, Paris., 1975.

Jean-Paul Vinay and Jean Darbelnet. *Stylistique comparée du français et de l'anglais*. Didier, 1958.

Jacob Wackernagel. Über ein Gesetz der indogermanischen Wortstellung. *Indogermanische Forschungen*, 1:333–436, 1892. ISSN 0019-7262. doi: 0.1515/9783110242430.333.

A. Waibel, A. N. Jain, A. E. McNair, H. Saito, A.G. Hauptmann, and J. Tebelskis. JANUS: A Speech-to-Speech Translation System using Connectionist and Symbolic Processing Strategies. In *Proceedings of the 1991 International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 793–796, 1991.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, arXive/1609.08144, 2016. URL `http://arxiv.org/abs/1609.08144`.

Yang You, Igor Gitman, and Boris Ginsburg. Scaling SGD Batch Size to 32K for ImageNet Training. *CoRR*, arXiv/1708.03888, 2017. URL `http://arxiv.org/abs/1708.03888`.

Zdeněk Žabokrtský, Jan Ptáček, and Petr Pajas. TectoMT: Highly modular MT system with tectogrammatics used as transfer layer. In *ACL 2008 WMT: Proceedings of the Third Workshop on Statistical Machine Translation*, pages 167–170, Columbus, OH, USA, 2008. Association for Computational Linguistics. ISBN 978-1-932432-09-1.

Zdeněk Žabokrtský, Jan Ptáček, and Petr Pajas. TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer. In *Proceedings of the 3rd Workshop on Statistical Machine Translation, ACL*, pages 167–170, 2008. URL `http://aclweb.org/anthology/W08-0325.pdf`.

D. Zeman. Reusable Tagset Conversion Using Tagset Drivers. In *Proceedings of LREC*, pages 213–218, 2008.

Daniel Zeman, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. HamleDT: To Parse or Not to Parse? In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 2735–2741, İstanbul, Turkey, 2012. European Language Resources Association. ISBN 978-2-9517408-7-7. URL `http://www.lrec-conf.org/proceedings/lrec2012/pdf/429_Paper.pdf`.

Daniel Zeman, Ondřej Dušek, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. HamleDT: Harmonized multi-language dependency treebank. *Language Resources and Evaluation*, 48(4):601–637, 2014. ISSN 1574-020X. URL `https://link.springer.com/article/10.1007/s10579-014-9275-2`.

Biao Zhang, Deyi Xiong, and Jinsong Su. Accelerating Neural Transformer via an Average Attention Network. *CoRR*, arXiv/1805.00631, 2018. URL http://arxiv.org/abs/1805.00631.

Yue Zhang and Joakim Nivre. Transition-based Dependency Parsing with Rich Non-local Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-2033.

Arnold M. Zwicky. On clitics. Technical report, Indiana University Linguistics Club, 1977. URL https://web.stanford.edu/~zwicky/on_clitics.pdf.

List of Abbreviations

ACC	accuracy
ADV	adverb
API	application programming interface
ASCII	American Standard Code for Information Interchange
BLEU	an automatic MT metric (bilingual evaluation understudy) [Papineni et al., 2002]
BPE	byte-pair encoding subword unites [Sennrich et al., 2016b]
CoNLL	Conference on Natural Language Learning
CS	ISO 639-1 language code for Czech
CUDA	Compute Unified Device Architecture
CZ	ISO 3166 country code for Czechia
DA	direct assessment manual evaluation method [Graham et al., 2016]
DAT	dative
EN	ISO 639-1 language code for English
FGD	Functional generative description [Sgall et al., 1986]
GB, GiB	gigabyte, gibibyte
GPU	graphics processing unit
GRU	gated recurrent unit [Cho et al., 2014]
HMTM	Hidden Markov Tree Model
IT	information technology
IWSLT	International Workshop on Spoken Language Translation
JHU	Johns Hopkins University
LM	language model
LSTM	long short-term memory [Hochreiter and Schmidhuber, 1997]
MERT	minimum error rate training [Och, 2003]
MT	machine translation
NLP	natural language processing
NMT	neural machine translation
OAA	one-against-all reduction
OOM	out-of-memory error
PCEDT	Prague Czech-English Dependency Treebank [Cuřín et al., 2004, Hajič et al., 2012]
PDT	Prague Dependency Treebank [Hajič et al., 2006]
PoS	part of speech
PRON	pronoun
QTLeap	European project `http://qtleap.eu/`
RBMT	rule-based machine translation
RNN	recurrent neural network
ROOT	special label assigned to a dependency-tree root
SGD	Stochastic Gradient Descent
SMT	statistical machine translation
TM	translation model
UD	Universal Dependencies `http://universaldependencies.org/`
Udapi	Universal Dependencies API `http://udapi.github.io/`
ÚFAL	Institute of Formal and Applied Linguistics (Charles University)
VERB	verb
WMT	Workshop/Conference on (Statistical) Machine Translation

List of Publications

Eneko Agirre, Arantxa Otegi, Gorka Labaka, Chakaveh Saedi, João Rodrigues, João Silva, Martin Popel, Roman Sudarikov, Kiril Simov, and Petya Osenova. Report on mt improved with semantic linking and resolving. Technical Report Deliverable D5.11, QTLeap Project, 2016. URL http://qtleap.eu/wp-content/uploads/2016/11/QTLEAP-2016-D5.11.pdf.

Nora Aranberri, Eleftherios Avramidis, Aljoscha Burchardt, Ondřej Klejch, Martin Popel, and Maja Popović. Tools and guidelines for principled machine translation development. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1877–1882, Paris, France, 2016. European Language Resources Association. ISBN 978-2-9517408-9-1.

Ondřej Bojar, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. The Joy of Parallelism with CzEng 1.0. In *Proceedings of the Eighth International Language Resources and Evaluation Conference (LREC'12)*, pages 3921–3928, Istanbul, Turkey, May 2012. ELRA, European Language Resources Association. ISBN 978-2-9517408-7-7.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, number 9924 in Lecture Notes in Artificial Intelligence, pages 231–238. Masaryk University, Springer International Publishing, 2016a. ISBN 978-3-319-45509-9.

Ondřej Bojar, Miloš Ercegovčević, Martin Popel, and Omar Zaidan. A Grain of Salt for the WMT Manual Evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 1–11, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2101.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2301.

Ondřej Dušek, Eva Fučíková, Jan Hajič, Martin Popel, Jana Šindlerová, and Zdeňka Urešová. Using Parallel Texts and Lexicons for Verbal Word Sense Disambiguation. In Eva Hajičová and Joakim Nivre, editors, *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 82–90, Uppsala, Sweden, 2015a. Uppsala University, Uppsala University. ISBN 978-91-637-8965-6.

Ondřej Dušek, Luís Gomes, Michal Novák, Martin Popel, and Rudolf Rosa. New Language Pairs in TectoMT. In *Proceedings of the 10th Workshop on Machine Translation*, pages 98–104, Stroudsburg, PA, USA, 2015b. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-941643-32-7. URL http://www.emnlp2015.org/proceedings/WMT/pdf/WMT09.pdf.

Ondřej Dušek, Zdeněk Žabokrtský, Martin Popel, Martin Majliš, Michal Novák, and David Mareček. Formemes in English-Czech Deep Syntactic MT. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 267–274, Montréal, Canada, 2012. Association for Computational Linguistics.

Ondřej Dušek, Martin Popel, Vivien Macketanz, Aljoscha Burchardt, Eleftherios Avramidis, Gertjan van Noord, João Rodrigues, António Branco, Gorka Labaka, Kiril Simov, and Aleksandar Popov. Report on the third mt pilot and its evaluation. Technical Report Deliverable D2.11, QTLeap Project, 2016. URL http://qtleap.eu/wp-content/uploads/2016/11/QTLEAP-2016-D2.11.pdf.

Rosa Del Gaudio, Aljoscha Burchardt, António Branco, and Martin Popel. Report on the embedding and evaluation of the third mt pilot. Technical report, QTLeap Project, 2016. URL http://qtleap.eu/wp-content/uploads/2016/11/QTLEAP-2016-D3.12.pdf.

Ondřej Klejch, Eleftherios Avramidis, Aljoscha Burchardt, and Martin Popel. MT-ComparEval: Graphical evaluation interface for Machine Translation development. *The Prague Bulletin of Mathematical Linguistics*, 104:63–74, October 2015. ISSN 0032-6585. doi: 10.1515/pralin-2015-0014. URL http://ufal.mff.cuni.cz/pbml/104/art-klejch-et-al.pdf.

David Mareček, Martin Popel, and Zdeněk Žabokrtský. Maximum Entropy Translation Model in Dependency-Based MT Framework. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 201–201, Uppsala, Sweden, 2010. Uppsala Universitet, Association for Computational Linguistics. ISBN 978-1-932432-71-8. URL http://www.aclweb.org/anthology/W10-1730.

Pavel Pecina, Ondřej Dušek, Lorraine Goeuriot, Jan Hajič, Jaroslava Hlaváčová, Gareth Jones, Liadh Kelly, Johannes Leveling, David Mareček, Michal Novák, Martin Popel, Rudolf Rosa, Aleš Tamchyna, and Zdeňka Urešová. Adaptation of machine translation for multilingual information retrieval in medical domain. *Artificial Intelligence in Medicine*, 61(3):165–185, 2014. ISSN 0933-3657.

Martin Popel and Ondřej Bojar. Training Tips for the Transformer Model. *The Prague Bulletin of Mathematical Linguistics*, 110:43–70, April 2018. ISSN 0032-6585. doi: 10.2478/pralin-2018-0002. URL https://ufal.mff.cuni.cz/pbml/110/art-popel-bojar.pdf.

Martin Popel and David Mareček. Perplexity of n-gram and dependency language models. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue. 13th International Conference, TSD 2010, Brno, Czech Republic, September 6-10, 2010. Proceedings*, volume 6231 of *Lecture Notes in Computer Science*, pages 173–180, Berlin / Heidelberg, 2010. Masarykova univerzita, Springer. ISBN 978-3-642-15759-2. doi: https://doi.org/10.1007/978-3-642-15760-8_23. URL `http://ufal.mff.cuni.cz/~popel/papers/2010_tsd.pdf`.

Martin Popel and Zdeněk Žabokrtský. Improving English-Czech Tectogrammatical MT. *The Prague Bulletin of Mathematical Linguistics*, 92:1–20, 2009. ISSN 0032-6585.

Martin Popel and Zdeněk Žabokrtský. TectoMT: Modular NLP Framework. *Advances in Natural Language Processing*, pages 293–304, 2010. URL `http://ufal.mff.cuni.cz/~popel/papers/2010_icetal.pdf`.

Martin Popel, David Mareček, Nathan Green, and Zdeněk Žabokrtský. Influence of Parser Choice on Dependency-Based MT. In Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan, editors, *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 433–439, Edinburgh, UK, 2011. University of Edinburgh, Association for Computational Linguistics. ISBN 978-1-937284-12-1. URL `http://www.aclweb.org/anthology/W11-2153`.

Martin Popel, Jaroslava Hlaváčová, Ondřej Bojar, Ondřej Dušek, António Branco, Luís Gomes, João Rodrigues, Andreia Querido João Silva, Nuno Rendeiro, Marisa Campos, Diana Amaral, Eleftherios Avramidis, Aljoscha Burchardt, Maja Popovic, Arle Lommel, Iliana Simova, Nora Aranberri, Gorka Labaka, Gertjan van Noord, Rosa Del Gaudio, Michal Novák, Rudolf Rosa, Aleš Tamchyna, and Jan Hajič. Report on the first mt pilot and its evaluation. Technical Report Deliverable D2.4, Version 1.8, QTLeap Project, 2015. URL `http://qtleap.eu/wp-content/uploads/2015/04/QTLEAP-2015-D2.4.pdf`.

Martin Popel, Zdeněk Žabokrtský, and Martin Vojtek. Udapi: Universal API for Universal Dependencies. In *NoDaLiDa 2017 Workshop on Universal Dependencies*, pages 96–101, Göteborg, Sweden, 2017. Göteborgs universitet. ISBN 978-91-7685-501-0.

Rudolf Rosa, Jan Mašek, David Mareček, Martin Popel, Daniel Zeman, and Zdeněk Žabokrtský. HamleDT 2.0: Thirty Dependency Treebanks Stanfordized. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, and Joseph Mariani, editors, *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 2334–2341, Reykjavík, Iceland, 2014. European Language Resources Association. ISBN 978-2-9517408-8-4. URL `http://www.lrec-conf.org/proceedings/lrec2014/pdf/915_Paper.pdf`.

Rudolf Rosa, Ondřej Dušek, Michal Novák, and Martin Popel. Translation model interpolation for domain adaptation in tectoMT. In Jan Hajič and António Branco, editors, *Proceedings of the 1st Deep Machine Translation Workshop*, pages 89–96, Praha, Czechia, 2015. ÚFAL MFF UK, ÚFAL MFF UK. ISBN 978-80-904571-7-1. URL `http://www.aclweb.org/anthology/W15-5711`.

Rudolf Rosa, Roman Sudarikov, Michal Novák, Martin Popel, and Ondřej Bojar. Dictionary-based domain adaptation of mt systems without retraining. In *Proceedings* *of the First Conference on Machine Translation*, pages 449–455, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2334.

Roman Sudarikov, Martin Popel, Ondřej Bojar, Aljoscha Burchardt, and Ondřej Klejch. Using MT-ComparEval. In Ondřej Bojar, Aljoscha Burchardt, Christian Dugast, Marcello Federico, Josef Genabith, Barry Haddow, Jan Hajič, Kim Harris, Philipp Koehn, Matteo Negri, Martin Popel, Georg Rehm, Lucia Specia, Marco Turchi, and Hans Uszkoreit, editors, *Translation Evaluation: From Fragmented Tools and Data Sets to an Integrated Ecosystem*, pages 76–82, Portorož, Slovenia, 2016. LREC. URL http://www.cracking-the-language-barrier.eu/wp-content/uploads/Sudarikov-etal.pdf.

Aleš Tamchyna, Martin Popel, Rudolf Rosa, and Ondrej Bojar. CUNI in WMT14: Chimera Still Awaits Bellerophon. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 195–200, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3322.

Daniel Zeman, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. HamleDT: To Parse or Not to Parse? In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 2735–2741, İstanbul, Turkey, 2012. European Language Resources Association. ISBN 978-2-9517408-7-7. URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/429_Paper.pdf.

Daniel Zeman, Ondřej Dušek, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. HamleDT: Harmonized multi-language dependency treebank. *Language Resources and Evaluation*, 48(4):601–637, 2014. ISSN 1574-020X. URL https://link.springer.com/article/10.1007/s10579-014-9275-2.

Appendix A

Author's Contribution

Prď na LaTeX, piš to normálně ve Wordu.
Markéta Popelová Nečasová

TectoMT-related software and publications

I have been the main developer of TectoMT and Treex since 2011. Detailed information about authorship of each component is available in the documentation and git/svn commit history.[1] As for the code discussed in more detail in this thesis:

- I implemented and evaluated the sentence-chunk parsing technique mentioned in Section 3.1 and Popel et al. [2011].
- The TectoMT code for reoredering of Czech clitics mentioned in Section 3.2 (block `T2A::CS::MoveCliticsToWackernagel`) was originally implemented by Zdeněk Žabokrtský, but refactored and substantially improved by myself.
- The MaxEnt TM (Section 3.3) was originally implemented by Zdeněk Žabokrtský and refactored and improved by myself and David Mareček.
- I am the sole author of the VowpalWabbit TM in TectoMT (Section 3.4). I contributed also to the VowpalWabbit framework (most importantly, adding the `--probabilities` option and refactoring of `--csoaa_ldf`).[2]
- In "Using Parallel Texts and Lexicons for Verbal Word Sense Disambiguation" [Dušek et al., 2015a], I contributed the VowpalWabbit model (including tuning and evaluation on a given task), while reusing the training data prepared by Ondřej Dušek.
- I redesigned the TectoMT analysis phase by extracting code common to multiple languages into base classes and improving it. I implemented the initial version of English↔Spanish and English↔Portuguese TectoMT and consulted the development of the final version, also for the other languages mentioned in Section 3.6.

[1] https://github.com/ufal/treex/commits?author=martinpopel
https://svn.ms.mff.cuni.cz/trac/tectomt_devel (username=password="public")

[2] https://github.com/JohnLangford/vowpal_wabbit/commits?author=martinpopel

- I conducted the experiments with VowpalWabbit TM fine-tuning (Section 3.7). The other reported domain-adaptation techniques were a joint work with the coauthors of Rosa et al. [2016, 2015].

NMT software and publications

- I contributed minor code improvements and bugfixes to T2T.[3] For example, adding English-Czech datasets or BLEU evaluation and helper scripts (`t2t-bleu`, `t2t-translate-all`).
- In "Training Tips for the Transformer Model" [Popel and Bojar, 2018], I designed, analyzed and described all the experiments (Chapter 4). Ondřej Bojar contributed to the sections Introduction (which is omitted in Chapter 4), Evaluation Methodology and Comparison with WMT17 Systems (which is rewritten in Chapter 4).
- I conducted all the experiments in Chapter 5.
- I prepared the English→Czech and Czech→English Transformer system submitted to WMT18. I thank Rico Sennrich, David Mareček and Jindřich Helcl for providing me with Nematus 2016 models and/or translations of NewsCrawl data by these models. I thank Dušan Variš and Ondřej Košarko for integrating the Transformer translation into Lindat services.

Other

- I wrote most parts of the two papers [Klejch et al., 2015, Sudarikov et al., 2016] about MT-ComparEval[4] and contributed to a third paper [Aranberri et al., 2016]. I implemented a prototype of the toolkit and supervised the final implementation done by Ondřej Klejch. The other co-authors of the papers implemented additional metrics (Hjerson, TER) and integration into `https://www.translate5.net`.
- I wrote "Udapi: Universal API for Universal Dependencies" [Popel et al., 2017]. The design of Udapi core was a joint work with Zdeněk Žabokrtský, who also implemented a first Python prototype. I did the Perl implementation of Udapi and re-implemented and improved the Python Udapi, contributing all the *blocks* described in the paper. Martin Vojtek did the Java implementation based on our specification of the API.
- In "Adaptation of machine translation for multilingual information retrieval in the medical domain" [Pecina et al., 2014], I did the automatic evaluation of MT (§ 2.4), helped with preprocessing (esp. English lemmatization) and contributed to the summary of related work (§ 2.1).

[3] `https://github.com/tensorflow/tensor2tensor/commits?author=martinpopel`

[4] `http://mt-compareval.ufal.cz/`

Appendix B

System Details

B.1 Translation Examples

T2009 and T2018 is TectoMT in versions 2009 and 2018 and “Our” is our final version of Transformer. Google, Bing and Yandex are translations obtained in July 2018 from the respective online translation services.[1]

source	TectoMT is currently an experimental system, which is outperformed by state-of-the-art MT systems such as open source Moses.
T2009	TectoMT je nyní experimentální systém, který je překonán state-of-the-art MT systémy otevřených zdrojových Mojžíšů.
T2018	Tectomt je nyní experimentální systém, který je překonán známými MT systémy jako otevření zdroj Mojžíši.
Google	TectoMT je v současné době experimentální systém, který je překonán nejmodernějšími systémy MT, jako je open source Moses.
Our	TectoMT je v současné době experimentální systém, který je překonán nejmodernějšími systémy MT, jako je open source Moses.
source	Birds of a feather flock together.
T2009	Ptáci v bederním hejnu spolu.
T2018	Ptáci péřového hejna spolu.
Google	Vrána k vráně sedá.
Our	Vrána k vráně sedá.
source	Great talkers are little doers.
T2009	Velcí řečníci jsou malí vrazi.
T2018	Velcí řečníci jsou malí lidé.
Google	Velcí mluvčí jsou malí lidé.
Our	Velcí mluvkové jsou malí dříči.
source	As good be an addled egg as an idle bird.
T2009	Dobré je feťácké vejce jako činný pták.
T2018	Dobří buďte plete vejce jako nečinný pták.
Google	Jako dobrá být včleněná vejce.
Our	Stejně dobré je být pomateným vejcem jako zahálejícím ptákem.
source	A miss by an inch is a miss by a mile.
T2009	Slečna palec je slečna miliónu.
T2018	Slečna palce je slečna míle.
Google	Chybějící palcem je míle vzdálená míle.
Our	Minutí o centimetr je o kilometr.

[1] Google Translate https://translate.google.com/; Bing Translator https://www.bing.com/Translator; Yandex Translate https://translate.yandex.com

source	I'd rather be a hammer than a nail.
T2009	Spíše bych byl kladivo než nehet.
T2018	Spíše bych byl kladivo než hřebík.
Google	Rád bych byl kladivo než hřebík.
Our	Radši budu kladivo než hřebík.
source	A bird in the hand is worth two in the bush.
T2009	Pták v ruce je cenný dvakrát v Bushovi.
T2018	Pták v ruce má hodnotu dva v buši.
Google	Pták v ruce stojí za dva v křoví.
Our	Pták v ruce má cenu dvou v buši.
source	Bread is the staff of life.
T2009	Chléb je zaměstnanec života.
T2018	Chléb je zaměstnanci života.
Google	Chléb je zaměstnancem života.
Our	Chléb je holí života.
source	For every kind of beasts, and of birds, and of serpents, and of things in the sea, is tamed, and *hath been tamed of mankind*: But the tongue can no man tame; it is an unruly evil, full of deadly poison.
T2018	Každého druhu zvířat a ptáků a hadů a věcí v moři je zkrocení a hath byl zkrocení lidstva: Ale jazyk nemůže žádný muž **tame**; je to **nevzpurné** zlo, plné smrtelného jedu.
Google	Pro každý druh dobytka, ptáků, hadů a všeho v moři je zkroucený a ponižován lidstvem. Ale jazyk nemůže splácet nikdo; je to neupřímné zlo, plné smrtícího jedu.
Our	**Neboť** každá šelma, ptáci, hadi a věci v moři jsou zkroceny a **lidstvo zkroceno jest**: Jazyk však člověk zkrotit nedokáže; je to nezkrotné zlo, plné smrtícího jedu.
refB21	Lidstvo se pokouší zkrotit každý druh zvířat, ptáků, plazů i mořských tvorů a daří se mu to, ale jazyk žádný člověk zkrotit neumí. Je to nezvládnutelné zlo, plné smrtelného jedu.
refČEP	Všechny druhy zvířat i ptáků, plazů i mořských živočichů mohou být a jsou kroceny člověkem, ale jazyk neumí zkrotit nikdo z lidí. Je to zlo, které si nedá pokoj, plné smrtonosného jedu.
refBKR	Všeliké zajisté přirození i zvěři, i ptactva, i hadů, i mořských potvor bývá zkroceno, a jest okroceno od lidí; Ale jazyka žádný z lidí zkrotiti nemůže; tak jest nezkrotitelné zlé, pln jsa jedu smrtelného.
source	Chytal tlouště na višni.
T2008	He caught chub to cherry.
Yandex	Catching chub on a cherry.
Bing	He caught chub on the višni.
Google	He caught the fat on the cherry.
Our	He was getting fat on the cherry.

source	Prď na LaTeX, piš to normálně ve Wordu.
T2018	Fart to LaTeX, write this normally in Word.
Bing	Prď on LaTeX, write it normally in Word.
Google	Go to LaTeX, write it normally in Word.
Our	Screw LaTeX, type it normally in Word.[2]

[2] I feel this is an appropriate place where to explain the abundance of footnotes in this thesis.

source	T2018	Yandex	Bing	Google	Our
I want translations with copious footnotes, footnotes reaching up like skyscrapers to the top of this or that page so as to leave only the gleam of one textual line between commentary and eternity. (Vladimir Vladimirovich Nabokov)	Chci překlady s hojnými poznámkami poznámky natáhnout se jako mrakodrapy k vrcholu tohoto nebo té stránky, tak nechte pouze lesk jedné textové čáry mezi komentářem a věčností. (Vladimir Vladimirovič Nabokov)	Chci, překlady s velkým množstvím poznámek pod čarou, poznámky pod čarou, dosahující až jako mrakodrapy na vrcholu tohoto, nebo tu stránku tak, aby se nechat jen záblesk jeden textový řádek mezi komentářem a věčnosti. (Vladimir Vladimirovič Nabokov)	Chci překlady s velkým poznámky pod čarou, poznámky pod čarou dosahující jako mrakodrapy na vrchol této nebo stránky tak, aby opustit pouze záblesk jedné textové čáry mezi komentářem a věčnost. (Vladimír Vladimirovič Nabokov)	Chci překlady srozsáhlými poznámkami pod čarou, poznámky pod čarou, které se dostanou nahoru jako mrakodrapy na vrchol této nebo té stránky, aby zanechaly jen záblesk jedné textové čáry mezi komentářem a věčností. (Vladimir Vladimirovich Nabokov)	Chci překlady s hojnými poznámkami pod čarou, s poznámkami pod čarou sahajícími jako mrakodrapy až nahoru na tu či onu stránku, aby zůstal jen záblesk jedné textové čáry mezi komentářem a věčností. (Vladimir Vladimirovič Nabokov)
Translation is like a woman. If it is beautiful, it is not faithful. If it is faithful, it is most certainly not beautiful. (Yevgeny Aleksandrovich Yevtushenko)	Překlad je jako žena. Pokud je krásné, není věrné. Pokud je věrné, není určitě krásné. (Yevgeny Alexandrovič Yevtushenko)	Překlad je jako žena. Pokud to je krásný, není věrný. Pokud je věřící, to je nejvíce rozhodně není krásné. (Jevgenij Alexandrovič Yevtushenko)	Překlad je jako žena. Pokud je to krásné, není to věrné. Pokud je věrný, je to určitě není krásná. (Jevgenij Alexandrovič Yevtushenko)	Překlad je jako žena. Pokud je to krásné, není to věrné. Pokud je věrná, určitě není krásná. (Jevgenij Aleksandrovič Yevtushenko)	Překlad je jako žena. Je-li krásná, není věrná. Je-li věrný, rozhodně není krásný. (Jevgenij Alexandrovič Jevtušenko)

B.2 TectoMT English-Czech Scenario

The following scenario listing can be obtained with `treex --dump Scen::EN2CS`.

```
Util::SetGlobal language=en selector=src
W2A::EN::Tokenize
W2A::EN::NormalizeForms
W2A::EN::FixTokenization
W2A::EN::TagMorce
W2A::EN::FixTags
W2A::EN::FixTagsImperatives
W2A::EN::Lemmatize
A2N::EN::NameTag
A2N::EN::DistinguishPersonalNames
W2A::MarkChunks
W2A::EN::ParseMST
 model=conll_mcd_order2_0.01.model
W2A::EN::SetIsMemberFromDeprel
W2A::EN::RehangConllToPdtStyle
W2A::EN::FixNominalGroups
W2A::EN::FixIsMember
W2A::EN::FixAtree
W2A::EN::FixMultiwordPrepAndConj
W2A::EN::FixDicendiVerbs
W2A::EN::SetAfunAuxCPCoord
W2A::EN::SetAfun
W2A::FixQuotes
W2A::EN::MarkCheckCommas
A2A::ConvertTags input_driver=en::penn
A2A::EN::EnhanceInterset
A2T::EN::MarkEdgesToCollapse
A2T::EN::MarkEdgesToCollapseNeg
A2T::BuildTtree
A2T::SetIsMember
A2T::EN::MoveAuxFromCoordToMembers
A2T::ProjectSelectedWild
A2T::EN::FixTlemmas
A2T::EN::SetCoapFunctors
A2T::EN::FixEitherOr
A2T::EN::FixHowPlusAdjective
A2T::FixIsMember
A2T::EN::MarkClauseHeads
A2T::EN::SetFunctors
A2T::EN::MarkInfin
A2T::EN::MarkRelClauseHeads
A2T::EN::MarkRelClauseCoref
A2T::EN::MarkDspRoot
A2T::MarkParentheses
A2T::SetNodetype
A2T::EN::SetFormemeInterset
A2T::EN::SetTense
A2T::EN::SetGrammatemes
A2T::SetGrammatemesFromAux
A2T::EN::SetSentmod
A2T::EN::RehangSharedAttr
A2T::EN::SetVoice
A2T::EN::FixImperatives
A2T::EN::SetIsNameOfPerson
A2T::EN::SetGenderOfPerson
A2T::EN::AddCorAct
T2T::SetClauseNumber
A2T::EN::FixRelClauseNoRelPron
A2T::EN::MarkReferentialIt
 resolver_type=nada threshold=0.5 suffix=nada_0.5
A2T::EN::FindTextCoref
Util::SetGlobal language=cs selector=tst
T2T::CopyTtree source_language=en source_selector=src
T2T::ProjectSelectedWild
T2T::EN2CS::TrLFPhrases
T2T::EN2CS::DeleteSuperfluousTnodes
T2T::EN2CS::TrFTryRules
T2T::EN2CS::TrFAddVariantsInterpol
 model_dir=data/models/translation/en2cs
 maxent_features_version=0.9
 models='
  static 1.0 formeme_czeng09.static.pls.slurp.gz
  maxent 0.5 formeme_czeng09.maxent.compact.pls.slurp.gz'
T2T::EN2CS::TrFRerank2
T2T::EN2CS::TrLTryRules
T2T::EN2CS::TrLPersPronIt
T2T::EN2CS::TrLPersPronRefl
T2T::EN2CS::TrLHackNNP
T2T::EN2CS::TrLAddVariantsInterpol models='
 static 0.5 tlemma_czeng09.static.pls.slurp.gz
 maxent 1.0 tlemma_czeng12.maxent.10000.100.2_1.compact.pls.gz
 static 0.1 tlemma_humanlex.static.pls.slurp.gz'
T2T::EN2CS::TrLFNumeralsByRules
T2T::EN2CS::TrLFilterAspect
T2T::EN2CS::TransformPassiveConstructions
T2T::EN2CS::PrunePersonalNameVariants
T2T::EN2CS::RemoveUnpassivizableVariants
T2T::EN2CS::TrLFCompounds
T2T::CutVariants
 lemma_prob_sum=0.5   max_lemma_variants=7
 formeme_prob_sum=0.9 max_formeme_variants=7
T2T::RehangToEffParents
T2T::EN2CS::TrLFTreeViterbi
T2T::RehangToOrigParents
T2T::CutVariants max_lemma_variants=3 max_formeme_variants=3
T2T::EN2CS::FixTransferChoices
T2T::EN2CS::ReplaceVerbWithAdj
T2T::EN2CS::DeletePossPronBeforeVlastni
T2T::EN2CS::TrLFemaleSurnames
T2T::EN2CS::AddNounGender
T2T::EN2CS::MarkNewRelClauses
T2T::EN2CS::AddRelpronBelowRc
T2T::EN2CS::ChangeCorToPersPron
T2T::EN2CS::AddPersPronBelowVfin
T2T::EN2CS::AddVerbAspect
T2T::EN2CS::FixDateTime
T2T::EN2CS::FixGrammatemesAfterTransfer
T2T::EN2CS::FixNegation
T2T::EN2CS::MoveAdjsBeforeNouns
T2T::EN2CS::MoveGenitivesRight
T2T::EN2CS::MoveRelClauseRight
T2T::EN2CS::MoveDicendiCloserToDsp
T2T::EN2CS::MovePersPronNextToVerb
T2T::EN2CS::MoveEnoughBeforeAdj
T2T::EN2CS::MoveJesteBeforeVerb
T2T::EN2CS::MoveNounAttrAfterNouns
T2T::EN2CS::FixMoney
T2T::EN2CS::FindGramCorefForReflPron
T2T::EN2CS::NeutPersPronGenderFromAntec
T2T::EN2CS::ValencyRelatedRules
T2T::SetClauseNumber
T2T::EN2CS::TurnTextCorefToGramCoref
T2T::EN2CS::FixAdjComplAgreement
Util::SetGlobal
 language=cs selector=tst source_language=en source_selector=src
T2A::CS::CopyTtree
T2A::CS::DistinguishHomonymousMlemmas
T2A::CS::ReverseNumberNounDependency
T2A::CS::InitMorphcat
T2A::CS::FixPossessiveAdjs
T2A::CS::MarkSubject
T2A::CS::ImposePronZAgr
T2A::CS::ImposeRelPronAgr
T2A::CS::ImposeSubjpredAgr
T2A::CS::ImposeAttrAgr
T2A::CS::ImposeComplAgr
T2A::CS::DropSubjPersProns
T2A::CS::AddPrepos
T2A::CS::AddSubconjs
T2A::CS::AddReflexParticles
T2A::CS::AddAuxVerbCompoundPassive
T2A::CS::AddAuxVerbModal
T2A::CS::AddAuxVerbCompoundFuture
T2A::CS::AddAuxVerbConditional
T2A::CS::AddAuxVerbCompoundPast
T2A::CS::AddClausalExpletivePronouns
T2A::CS::MoveQuotes
T2A::CS::ResolveVerbs
T2A::ProjectClauseNumber
T2A::AddParentheses
T2A::CS::AddSentFinalPunct
T2A::CS::AddSubordClausePunct
T2A::CS::AddCoordPunct
T2A::CS::AddAppositionPunct
T2A::CS::CheckCommas
T2A::CS::ChooseMlemmaForPersPron
T2A::CS::GenerateWordforms inflect_by_ending=1
T2A::CS::DeleteSuperfluousAuxCP
T2A::CS::MoveCliticsToWackernagel
T2A::CS::DeleteEmptyNouns
T2A::CS::VocalizePrepos
T2A::CS::CapitalizeSentStart
T2A::CS::CapitalizeNamedEntitiesAfterTransfer
A2A::ProjectCase
A2W::ConcatenateTokens
A2W::CS::ApplySubstitutions
A2W::CS::DetokenizeUsingRules
A2W::CS::RemoveRepeatedTokens
```