Enriching Neural MT through Multi-Task Training
I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

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

In Prague, 30 June 2018

signature of the author
I would like to express my sincere gratitude to Ondřej Bojar, the supervisor of this thesis, for his overall support during the last year, for providing feedback and consultations. I also thank to Martin Popel for his recommendations of the best practices for using the software. Many thanks belong to all ÚFAL cluster users, and especially to its administrator Milan Fučík, for understanding and patience with my very extensive allocation of computational resources. I also have to thank my mother, Marcela Macháčková, besides others, for manual evaluation of MT outputs. Finally I would like to express gratitude to my beloved family and parents in our mother language.

Děkuji svým rodičům a celé rodině za nesmírnou podporu, duchovní i materiální, za trpělivost a pochopení nejen v průběhu mého vysokoškolského studia.
Title: Enriching Neural MT through Multi-Task Training

Author: Dominik Macháček

Institute: Institute of Formal and Applied Linguistics

Supervisor: RNDr. Ondřej Bojar, PhD., Institute of Formal and Applied Linguistics

Abstract: The Transformer model is a very recent, fast and powerful discovery in neural machine translation. We experiment with multi-task learning for enriching the source side of the Transformer with linguistic resources to provide it with additional information to learn linguistic and world knowledge better. We analyze two approaches: the basic shared model with multi-tasking through simple data manipulation, and multi-decoder models. We test joint models for machine translation (MT) and POS tagging, dependency parsing and named entity recognition as the secondary tasks. We evaluate them in comparison with the baseline and with dummy, linguistically unrelated tasks. We focus primarily on the standard-size data setting for German-to-Czech MT. Although our enriched models did not significantly outperform the baseline, we empirically document that (i) the MT models benefit from the secondary linguistic tasks; (ii) considering the amount of training data consumed, the multi-tasking models learn faster; (iii) in low-resource conditions, the multi-tasking significantly improves the model; (iv) the more fine-grained annotation of the source as the secondary task, the higher benefit to MT.

Keywords: multi-task neural machine translation NMT Transformer German
# Contents

## Introduction

1 Theory Background

1.1 Machine Translation ........................................ 5

1.1.1 Phrase-Based versus Neural MT .......................... 5

1.2 Neural Machine Translation Models ........................ 7

1.2.1 RNN Encoder–Decoder .................................... 8

1.2.2 Attentional Encoder–Decoder Model ...................... 8

1.2.3 Transformer Model ....................................... 10

1.3 Linguistic Knowledge in MT .................................. 12

1.3.1 Treebanks and Supervised Learning ...................... 12

1.3.2 Universal Dependencies ................................. 13

1.3.3 Prague Dependency Treebank ............................. 14

2 Related Work

2.1 Enriching Source through Multi-Task NMT .................. 17

2.1.1 Multi-Task NMT Architectures ............................ 17

2.1.2 Task Identification ....................................... 19

2.1.3 Scheduled Multi-Task Learning ........................... 19

2.1.4 Discussion about General Applicability ................ 20

2.2 Enriching Target through Multi-Task NMT ................ 21

2.2.1 Interleaving vs Multi-Decoder in sequential model .... 21

2.2.2 Interleaving in Transformer .............................. 22

2.3 Enriching Source by Linguistic Features in Word Embeddings 23

3 Basic Settings for Experiments

3.1 German-to-Czech Dataset ................................. 25

3.1.1 Test and Validation ..................................... 25

3.1.2 Training .................................................. 25

3.2 NMT Frameworks .......................................... 27

3.2.1 Tensor2Tensor ........................................... 27

3.2.2 OpenNMT .................................................. 28

3.2.3 Training Tips for the Transformer Model ............... 28

3.2.4 Note on Tensor2Tensor Terminology .................... 29

3.3 Word Segmentation ...................................... 30

3.4 Automatic Evaluation ...................................... 30

4 Multi-Tasking through Simple Alternating ................. 33

4.1 Variable Parameters of Experiments ........................ 33

4.1.1 Multitasking Model Settings ............................. 33

4.1.2 Multitasking Variants ................................... 34

4.1.3 Data Settings ........................................... 34

4.1.4 Secondary Tasks ....................................... 36

4.1.5 Designing Vocabulary .................................. 42

4.2 Experiments with Simple Alternating Setup ............... 43
Introduction

Machine translation is widely used nowadays as a simple, fast and affordable tool for overcoming language barriers. The quality of machine translation achieved big improvements in last years, since efficient computer hardware and deep neural networks have been developed. Despite of it, machines still have severe limitations in language understanding. They are not able to translate arbitrary texts neither equally well nor better than skillful human translators. The state-of-the-art machine translation outputs are often very fluent, almost not distinguishable from human translations, but may not be always reliable. The content of the translated text may not correspond to the original meaning. In this work, we contribute to the research of machine translation by exploring new approaches, algorithms and techniques to include explicit linguistic knowledge in the training.

The main scope of this work is enriching neural machine translation (NMT) through multi-task learning, which has already shown promising results in various areas of deep learning applications. NMT has also already benefited from enriched information in either the source or the target language (e.g. part-of-speech tags, syntactic tags from CCG and others) for the sequence-to-sequence model with attention. We experiment with the newest state-of-the-art Transformer architecture, whose adaptation to multi-task learning seems to be very promising, but not yet sufficiently explored.

The Transformer architecture is relatively new. It is implemented in Tensor2Tensor framework [Vaswani et al., 2018] and several other toolkits (OpenNMT [Klein et al., 2017], Marian [Junczys-Dowmunt et al., 2018], Neural Monkey [Helcl and Libovický, 2017]). The primary and best-performing implementation is the Tensor2Tensor one, which however does not support multi-task learning. We therefore primarily explore a simple variant of multi-tasking with the basic model, where the multiple tasks are reflected only in the training data. This gives the community of Tensor2Tensor contributors reasons to either implement the support for multi-tasking with task-specific model components, if the simple multi-tasking does not perform well, or not implement it if the basic model provides it already.

Our experiments are carried primarily on German-to-Czech language pair, which is relatively challenging due to the rich morphology of both languages and also due to the limited research interest this language pair has received. The secondary goal of this work is to find the current state-of-the-art of German-to-Czech machine translation, and find the best possible settings.

Thesis Organization

In Chapter 1, we summarize the background of our work. We start with the conditions, in which the neural machine translation is best applicable. We survey neural machine translation models, their advantages and drawbacks. We also mention the treebanks, which can be used for human supervision of machines to acquire linguistic knowledge.

In Chapter 2, we analyze the works related to our research and discuss their general applicability.
In Chapter 3, we describe and give reasons for the basic settings for our experiments: the dataset, an implementation of the Transformer model, hyperparameter settings, and a method for subword segmentation.

Chapter 4 contains the most important contribution of our work, the experiments with multi-tasking through simple alternating, together with comparison of impacts of design decisions (auxiliary linguistic tasks, quality of training data, sampling of the multiple tasks, vocabulary design etc.) for their background. We manually evaluate the most promising runs and conclude with interesting findings.

In Chapter 5, we provide our results with implementation of the multi-tasking model with task-specific components in OpenNMT-py.

Finally, we summarize the whole work in Conclusion.
1. Theory Background

In this chapter we summarize the theory background used in this work. In Section 1.1, we briefly review the basic approaches to machine translation (MT), and compare two most recent ones: phrase-based and neural machine translation (NMT). The fact that NMT performs better, when it is trained on a big amount of data, gives us reasons to use all available training data for our further NMT experiments, although it is not convenient due to time complexity.

Section 1.2 reviews the NMT models and concludes with reasons, why to focus our research on the Transformer model.

Finally, in Section 1.3 we describe the necessity of linguistic and world knowledge in MT, and summarize supervised learning and treebanks.

1.1 Machine Translation

The main basic approaches to MT are rule-based, statistical (often abbreviated as SMT), and neural (NMT). The most widespread SMT technique is phrase-based machine translation (PBMT), which had been state-of-the-art until 2015, when the system of University of Montreal won WMT shared German-to-English translation task using NMT and obtained good results on other language pairs [Bojar et al., 2015]. NMT is also used in production in the commercial sphere.²

We advise a reader to follow “A history of machine translation from the Cold War to deep learning” blogpost by Pestov [2018]. Rule-based machine translation, its origin, advantages and drawbacks are described there in more details. Nowadays, rule-based machine translation may be used for low-resource languages, for which not enough training data are available to train phrase-based or neural MT models. For high-resource language pairs, it is usually easier to use phrase-based or neural MT, because such a model can be trained fully automatically from training data. No linguistic knowledge and work of experts to observe, define and formalize rules is necessary.

Zhang [2018] describe history and frontier of the NMT.

1.1.1 Phrase-Based versus Neural MT

Despite of being rated better in WMT shared tasks, NMT is not the best approach under certain conditions. In this section we compare it with PBMT.

When PBMT outperforms NMT

Koehn and Knowles [2017] describe six challenges of NMT which PBMT systems solve better. We select following:

- domain mismatch – when a system is trained on different data than the user asks to translate, NMT may output a completely fluent but unrelated

¹Strictly speaking, the neural MT is also statistical, but it is so different and specific that it is not considered as SMT.
²See https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html for an evidence that Google Translate uses NMT.
sentence. The user may not notice it, and can be led to confusion and misunderstanding. PBMT instead produces usually an ungrammatical sentence where foreign words from the source side may be copied.

- **adequacy** – PBMT systems rarely omit important parts of the source sentence, like whole clauses, although it may have problems e.g. with negation and long range dependencies (like agreement of verb form depending on subject etc.). NMT sacrifices adequacy for fluency.

- **amount of training data** – NMT outperforms PBMT, only if large amount of training data are available. In English-to-Spanish scenario reported by the authors, NMT gained the same score as PBMT with 15M source tokens. With only 377k source tokens, PBMT system was substantially better. See Figure 1.1.

![BLEU Scores with Varying Amounts of Training Data](image)

Figure 1.1: BLEU scores for English-Spanish systems trained on 0.4 million to 385.7 million words of parallel data. Quality for NMT starts much lower, outperforms SMT at about 15 million words, and even beats a SMT system with a big 2 billion word in-domain language model under high-resource conditions. Figure and caption reprinted from Koehn and Knowles [2017].

- **long sentences** – PBMT may outperform NMT in sentences longer than 60 words. We assume it is because NMT model considers that any word in a sentence may be the translation of any other particular word. NMT models may need lots of long sentence examples to learn correspondence of words. The space of long sentences is huge and sparse. On the other hand, PBMT usually segments long sentences into shorter fragments and translates them separately.

- **interpretability of model behavior** – in PBMT, an expert might be able to predict the model performance and behavior from the hyperparameter settings. He can also rigorously explain any particular translation with statistics. The model parameters are stored as observations of particular
words and phrases. In NMT, and in deep learning generally, there are many hyperparameters affecting the model behavior, and the behavior is usually unpredictable from the hyperparameters. Neural network training is also nondeterministic, due to random initialization and strict dependence on corpus ordering and float rounding errors. NMT model parameters are very complex, represented with rational numbers, and particular NMT outputs are inexplicable nowadays.

When NMT outperforms PBMT

The conditions under which NMT outperforms PBMT, are:

- rare and unknown words – in PBMT, the number of words a model can recognize is unlimited and depends only on the training data. However, completely unknown words can not be handled any way, they can be only copied from source. Unlike PBMT, NMT model can learn similarities between similar rare words, recognize their morphological categories and inflect them (by sticking the wordpieces together), or learn to copy them from source.

- fluency – NMT outperforms PBMT in grammar. The unsolved issues of PBMT, double negation in languages like Czech, long range dependencies etc., cause little or no problems at all in NMT. Belinkov et al. [2017] empirically show that NMT models learn morphology. They also show that translating into morphologically rich languages is still challenging and more difficult than into not so morphologically rich ones.

1.2 Neural Machine Translation Models

Let us describe the translation problem from the mathematical point of view. Given two languages $E$ and $F$, which are sets of sentences, translation is a function $t : E \rightarrow F$, mapping from a sentence $x \in E$ to its equivalent $y \in F$.

In data-driven MT, we have a parallel corpus of sentences in source and target languages. We call it parallel because the sentences come in pairs, corresponding to each other. Each word, phrase and sentence, not only from the parallel corpus, has its probability of being translated to every other word, phrase or sentence. The actual probabilities are estimated from observations from the corpus. MT thus aims to find the most probable translation for a given source sentence $x$: $t(x) = \arg\max_y p(y|x)$.

In NMT, the argmax computation and probability estimation is provided by a deep neural network model.

In this work we further assume the reader is familiar with theory of artificial neural networks, including deep and recurrent NN (RNN). If not, we advise reader to follow Goodfellow et al. [2016].

In following sections we bring a quick overview of recent NMT models with their basic features, advantages and drawbacks. More details about them can be found in the referenced studies.
1.2.1 RNN Encoder–Decoder

The basic NMT model (by Sutskever et al. [2014]) contains two main components, both implemented with recurrent neural network (RNN).

NMT expects a fixed-vocabulary on the source side, its size can be e.g. 50,000. Input words are embedded from one-hot encoding to continuous-space word representation to a smaller vector of size e.g. 500.

Consecutively, an RNN (called encoder) encodes a variable length source sequence into a single fixed-sized context vector. The other RNN (decoder) decodes it into the target sentence. See Figure 1.2 for an illustration.


Cho et al. [2014] empirically confirmed limitations of this simple model. The translation quality decreases as the source sentence length increases, when the model size is small. It may be caused by the size limit of the context vector, and by exploding gradient issue in RNN. The total model size is restricted by memory limits.

1.2.2 Attentional Encoder–Decoder Model

Bahdanau et al. [2014] suggested a soft attention mechanism to address the issue with decreasing translation quality on increasing source sentence length. The encoder is constructed as a bidirectional RNN instead of the original single-directional one. The decoder receives not only the context vector and the previously translated word, but also learns to look at specific tokens in the source
sentence to recall the relevant context. This frees the model from having to encode the whole information contained in the sentence into a fixed size vector. Longer sentences are translated better than with the basic model of the same size. See the illustration in Figure 1.3.

$$f = \text{(La, croissance, économique, s'est, ralentie, ces, dernières, années, .)}$$

![Diagram](https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/)

Figure 1.3: Illustration of attentional encoder-decoder NMT model. Figure reprinted from https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/.

This architecture has still one drawback, the sequentiality. Encoding is linear and can not be parallelized. The sentence must be processed from left to right or right to left, although some sentences may benefit from starting the translation at another place.
1.2.3 Transformer Model

Vaswani et al. [2017] introduce a new model called Transformer. Its key idea is that only the attention is necessary for translation, while recurrence and sequentiality can be fully omitted. Attention is implemented with matrix multiplication, which is very fast on current hardware. Thanks to this, the model is incomparably faster in training phase. Representation of the absolute positions of words are provided by special positional encoding, so the model internally works with bags of words and features to consider the distances between words.

Figure 1.4 illustrates the Transformer architecture. The left part is the encoder, the right is part is the decoder. One attention component is inside the encoder. It assigns to each word an attention score of every other word in the sentence. The authors claim that this module is able to solve coreference between any pair of words in single step, while sequential models need a higher number of steps to transport the information to corresponding positions in sequence, and thus performs slower and worse. Another attention module is inside the decoder and on the boundary between the encoder and decoder.

Vaswani et al. [2017] show experiments that the Transformer architecture outperformed previous model with much lower training cost. In their blogpost, the authors claim (without further details) that the same network they used for English to German translation, with a little adaptation, outperformed all but one of the previously proposed approaches to constituency parsing.

These results are very promising, setting the new state-of-the-art. It also suggest further exploration of the Transformer model, which falls nicely into the main scope of our work.

---

Figure 1.4: The Transformer model architecture. Figure and caption reprinted from Vaswani et al. [2017].
1.3 Linguistic Knowledge in MT

A certain part of linguistic and world knowledge is necessary for translation, including automatic translation by machines. Let us illustrate it on the following sentence:  

A professor won an award. When he tried to pack it into his suitcase, he realized...  

– ...it was too big.  
– ...it was too small.

When we want to translate this sentences into e.g. German or Czech, we need to know, which entity is represented by the last occurrence of the pronoun it. It is award in the first variant and suitcase in the latter. Both award and suitcase have a different gender in German or Czech, which must be reflected in a correct translation.

This particular linguistic task is called coreference resolution, and it is one of many subtasks necessary for translation. However, to solve this particular example, the linguistic knowledge alone is not sufficient. The system really has to consult the world knowledge. A human has usually no issues with understanding and translating this sentence, but (current) computers have.

Figure 1.5 shows the Vauquois triangle [Vauquois, 1968]. It illustrates different approaches to machine translation hierarchized by the depth of formal linguistic analysis. Plain phrase-based MT is an example of direct translation, it operates on word level. Plain neural MT usually internally operates at a higher level than direct translation, although the input and output are again plain sequences of words or subwords. There are evidences that neural networks are able to learn some morphology and syntax on their own, only from observations from parallel data. It is assumed that neural networks are able to develop their own kind of interlingua, which has a form hidden states of the network, uninterpretable by humans. However, Figure 1.6 shows Czech-to-English example of NMT by Google Translate (as in June 2018), from which it is obvious that the model lacks deep understanding about entities mentioned in the sentence, so we assume its internal semantic structure is incomplete.

In this work, we focus on enriching NMT by deeper linguistic knowledge, either at level of morphology, syntax or semantics, which should support a neural network in deeper language understanding.

1.3.1 Treebanks and Supervised Learning

A treebank is a big collection of texts manually annotated by experts at a level of syntax and often also at lower levels. The most important treebanks use a structure of tree for syntax representation, therefore we call them “treebanks”.

An automatic tool to annotate raw text the same way as in the treebank can be trained with supervised learning from a manually annotated treebank. This is a way how a machine can acquire linguistic or any kind of knowledge.

---

4This example originates in the Winograd Schema Challenge [Levesque, 2011].
5See for example CoNLL 2017 Shared Task on parsing in many languages: http://universaldependencies.org/conll17/
It should be noted that by supervised learning, the machine can not outperform the teacher. Treebanks provide artificial linguistic information designed by human experts. We assume it is related to language understanding and helps machine translation, however, it is possible that the neural network finds its own way to cover the necessary knowledge. The supervision may force the network to perform well on the artificial task and construct answers in expected shapes, but at the same time it prevents the network from finding its own, possibly more efficient way to gain the knowledge with respect to MT.

We further provide more details about treebanks used in our work.

### 1.3.2 Universal Dependencies

Universal Dependencies (UD, Nivre et al. [2017]) is a collection of more than 100 treebanks for more than 60 languages created by open community of over 200 contributors. All the treebanks use a uniform annotation schema for morphology and dependency syntax.

In our work, we use UD for German. Each sentence is annotated with lemmas, universal part-of-speech (POS) tags, specific POS tags for the given language,
morphological features and values (e.g. case, number, person, mood etc.), and with a syntactic dependency tree. See illustration in Figure 1.7.

![Dependency Tree](image)

<table>
<thead>
<tr>
<th>Id</th>
<th>Form</th>
<th>UPosTag</th>
<th>XPosTag</th>
<th>Feats</th>
<th>Head</th>
<th>DepRel</th>
<th>Deps</th>
<th>Misc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ich</td>
<td>PRON</td>
<td></td>
<td>Case=Nom</td>
<td>Number=Sing</td>
<td>Person=1</td>
<td>PronType=Frs</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>habe</td>
<td>VERB</td>
<td>VAFIN</td>
<td>Mood=Ind</td>
<td>Number=Sing</td>
<td>Person=3</td>
<td>Tense=Pres</td>
<td>VerbForm=Fin</td>
</tr>
<tr>
<td>3</td>
<td>eine</td>
<td>DET</td>
<td>ART</td>
<td>Case=Ac</td>
<td>Definite=Ind</td>
<td>Number=Plur</td>
<td>PronType=Art</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Katze</td>
<td>NOUN</td>
<td>NN</td>
<td>Case=Acc</td>
<td>Gender=Fem</td>
<td>Number=Sing</td>
<td>2</td>
<td>SpaceAfterNo</td>
</tr>
<tr>
<td>5</td>
<td>.</td>
<td>PUNCT</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>SpaceAfterNo</td>
</tr>
</tbody>
</table>

Figure 1.7: German sentence *I have a cat.* annotated with Universal Dependencies. Visualization of syntactic dependency tree on top, table with all annotations at the bottom. Example provided by [http://lindat.mff.cuni.cz/services/udpipe/](http://lindat.mff.cuni.cz/services/udpipe/).

### 1.3.3 Prague Dependency Treebank

Prague Dependency Treebank (PDT, Bejček et al. [2011]) is a collection of Czech news texts annotated on 4 layers:

- **Word Layer (w-layer)** – raw text, space delimited words

- **Morphological Layer (m-layer)** – typing errors are corrected, conflation word-forms (e.g. Czech *pročs* → *proč jsí*, lit.: “why-are-you”) are separated. Each word has assigned a morphological lemma and morphological tag, which includes part-of-speech, gender, case, number, tense and many other features. PDT uses a positional tagset with 15 elements.

- **Analytical Layer (a-layer)** – each sentence is provided with its syntactic dependency tree. Each word in the sentence is a node, it has an analytical functor (a label in the dependency tree), dependency head (father) and dependent nodes (children).
- Tectogrammatical Layer (t-layer) – represents deep syntactic structure of a sentence. Each real-world entity mentioned in the sentence has a node in the t-tree. The node is annotated with the tectogrammatical lemma, tectogrammatical functor (predicate, object, subject, agent, patient etc.), reference to the corresponding nodes in a-tree, semantic part-of-speech etc.

See illustration on Figure 1.8.

Figure 1.8: Relations between layers in PDT. The rendered Czech sentence *Byl by šel dolesa.* (lit.: *He-was would went to forest.*) contains past conditional of the verb *jít* (to go) and a typo *dolesa* instead of *do lesa.* Figure and example reprinted from https://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/en/html/ch02.html.
2. Related Work

In this chapter, we review previous works related to our research of enriching NMT through multi-task training.

2.1 Enriching Source through Multi-Task NMT

One of the first works successfully utilizing multi-task training is Google’s multilingual NMT system [Johnson et al., 2017]. The basic attentional sequence-to-sequence model, the same architecture as for translation of one language pair, is trained to perform translations from many to many languages (or from one to many, or from many to one). All model components are shared and the neural network is thus able to generalize across language boundaries. The experiments show the network benefits from the sharing. Sharing model components also enables translation between language pairs, whose parallel data were not available at training, or rarely available. This is so called “zero-shot” translation. Better results are achieved, when the joint translation is performed on related languages.


All these works (Johnson et al. [2017], Niehues and Cho [2017], Zaremoodi and Haffari [2018], and Kiperwasser and Ballesteros [2018]) use basic attentional sequence-to-sequence NMT model, adapted to multi-tasking. Since the Transformer model is relatively new, its adaptation for multi-tasking is not yet implemented nor explored, although it seemed to be one of the intended goal in early releases of the Tensor2Tensor toolkit by Google.\(^1\)

We further analyze various aspects of the mentioned related works.

2.1.1 Multi-Task NMT Architectures

In this section, we summarize the approaches of adaptation of the attentional sequence-to-sequence model to perform multi-tasking.

All Components Shared

The first approach, used by Johnson et al. [2017], Kiperwasser and Ballesteros [2018], and Niehues and Cho [2017] (they further implement another methods for comparison), is to use the same model as for single-task MT. The advantage is that the model shares all components for all sides of thus has the capacity to generalize the components across task boundaries, and the network can benefit

\(^1\)The support for multi-tasking in T2T used to exist, but it did not reliably work and was removed in version 1.4.0 (December 2017). See https://github.com/tensorflow/tensor2tensor/issues/487
from task relatedness. However, this only enables the generalization, it does not force it.

This approach is desirable for multilingual MT and for scenarios, where all tasks have an equal importance and are relevant to each other. In MT with auxiliary task, where the auxiliary task target differs enough from MT, the overlap for formulating the auxiliary task output employs a certain capacity of the network, which then can not be used for the main task properly.

Since the same source can correspond to different targets for various tasks, it is necessary to provide the network the task identification. We analyze this in Section 2.1.2.

**Multi-Decoder Models**

Another approach is to use distinct model components, as proposed by Niehues and Cho [2017] and Zaremoodi and Haffari [2018]. Niehues and Cho [2017] attempted to answer a question, how much parameter sharing between tasks is desirable. In their settings they showed improvements of MT+POS and MT+POS+NER over baseline MT with all components shared, but better scores with distinct decoders and attention.

Similarly, Kiperwasser and Ballesteros [2018] compared model with shared encoder and decoder and model with shared encoder and distinct decoders. They obtained better results with the latter.

See the illustration from Niehues and Cho [2017] in Figure 2.1.

![Figure 2.1: Overview of different architectures for attentional sequence-to-sequence NMT multi-task learnings. Figure and caption reprinted from Niehues and Cho [2017].](image)

**Deep Stacked Task-Specific Components**

Zaremoodi and Haffari [2018], on the other hand, use a “deep stacked encoder and decoder”. These components consist of several stacked layers. If we have $n$
auxiliary tasks, then the \( n \)-th task utilizes the layers 1 to \( n \). This is desirable, if the tasks are of increasing complexity and the depth of language processing, as morphology, syntax, semantics and MT. There is an evidence from image recognition that the lower layers of deep neural networks cover the basic features (as e.g. edge detection), while the upper layers apply and combine them to recognize more complex patterns (e.g. faces, animals, etc.). This can be applied to NMT as well, as Zaremoodi and Haffari [2018] show.

The training objective in Zaremoodi and Haffari [2018] is adapted to multi-task learning. It is the sum of losses for all tasks, weighted by the inverted size of training data for the particular task and balancing parameters. In each training step, a training example for MT is selected randomly and paired with an example for one random auxiliary task. This prevents the training signal from the main task to be washed out by auxiliary tasks.

**Task Discriminator and Adversarial Learning**

Furthermore, Zaremoodi and Haffari [2018] implement a task discriminating component. It attempts to predict from the features based on some intermediate layers in the network, which task is currently being processed. On correct prediction, positive loss is added to the training objective. It forces the network to generalize its internal representation, abstracting away task specific components. This technique is called “Adversarial Learning” [Lowd and Meek, 2005].

### 2.1.2 Task Identification

For models with the shared decoder it is important to identify the task because the source can be identical for multiple tasks. Johnson et al. [2017] solve this problem simply by adding a special task identification token to the source. This approach does not prevent the encoder from building separate internal models for each task, which may be not desirable.

Kiperwasser and Ballesteros [2018] suggest prepending a task identification vector to the target side, which forces parameter sharing in the encoder, because it has no way to determine the intended task and adapt the output accordingly.

Zaremoodi and Haffari [2018] do not specify their approach to task identification, but in models with task specific output components, the trainer must be responsible for switching the signal flow into the specific component.

### 2.1.3 Scheduled Multi-Task Learning

Kiperwasser and Ballesteros [2018] explored the “scheduled multi-task learning”. If we have two tasks, then in each training step, the task, whose training example is going to be processed, is selected by random coin toss by “Scheduler”. The probability \( p(t) \) of selecting the main task in timestep \( t \) can vary over time. The rest of probability, \( 1 - p(t) \), is uniformly distributed to the remaining auxiliary tasks.

In “Constant Scheduler”, the probability remains constant, \( p(t) = \alpha \). In “Exponential Scheduler”, \( p(t) = 1 - e^{-\alpha t} \). It starts training with auxiliary tasks, and with continued training it focuses only on the main task. “Sigmoid Scheduler”
\( p(s) = \frac{1}{1 + e^{-\alpha s}} \) starts with proper multi-tasking, and as the training graduates, it finetunes only to the main task. See illustration in Figure 2.2.

![Illustration of different scheduling strategies](image)

**Figure 2.2:** Illustration of different scheduling strategies determining the probability of the next training example to be picked from each of the multiple tasks we learn. Each sub-plot in the figure matches a different scheduling strategy (with \( \alpha \) set to 0.5). The sub-plot describes the probability \( p \) (y-axis) of the task we wish to improve \( q \) using Scheduled Multi-Task Learning as a function of the number of epochs trained \( t \) (x-axis) so far. The remaining probability is uniformly distributed among the rest of the tasks. Figure and caption reprinted from Kiperwasser and Ballesteros [2018].

### 2.1.4 Discussion about General Applicability

So far, we have not mentioned the datasets, language pairs and amounts of data, on which the authors validate their results. Niehues and Cho [2017] and Zaremoodi and Haffari [2018] report new promising approach “especially for low-resource conditions” or for “bilingually scarce scenarios”. We have to be careful in generalizations on the applicability and usefulness of their approaches to other language pairs and datasets.

#### Datasets and Scores

Niehues and Cho [2017] use German-to-English WIT\(^3\) corpus [Cettolo et al., 2012] with 4M tokens (number of sentences unspecified). They report only a minor improvement, baseline MT 30.85 versus multi-task model adapted to MT 32.30 on tst2013 provided by International Workshop on Spoken Language Translation (IWSLT). On two other testsets, the improvement was even smaller.

Zaremoodi and Haffari [2018] validate results on translation from English to French, Vietnamese and Farsi, three languages with varying degrees of divergence from English. Their English-to-French dataset is a subsample of Europarl v7 [Koehn, 2005]. It has 99k\(^2\) sentence pairs and approximately 2.4M tokens.\(^4\) Their validation baseline BLEU (on newstest2012) is 8.85, with multi-tasking and adversarial training they received 11.93. The datasets and results for Farsi and Vietnamese are similar. Note that such low BLEU scores can be particularly unreliable [Bojar et al., 2010].

---

\(^2\) We assume this number represents the number of parallel sentences, but it could be also the number of tokens. Zaremoodi and Haffari [2018] do not specify it.

\(^3\) We estimated this number from average number of words in a sentence in Europarl v7.

\(^4\) We estimated this number from average number of words in a sentence in Europarl v7.
Kiperwasser and Ballesteros [2018] report experiments in two scenarios. In “low-resource”, they use the same German-to-English data as Niehues and Cho [2017]. In “standard” they use the WMT parallel corpus [Buck et al., 2014] with 4.5M sentence pairs, English-to-German. They use newstest2014 for testing. Maximum sentence length is 60. The best score with standard setting, NMT+POS with “Exponential Scheduler”, outperformed baseline MT by 0.8 BLEU (19.20 versus 20.02) while the same model did not outperform baseline POS tagger or parser.

Discussion

In low-resource conditions, the neural network probably can not learn much to make proper generalizations for translation (recall Section 1.1.1). Additional linguistic information may be beneficial because the encoder is provided with additional training examples related to the desired MT task. Providing additional parallel sentences to train the MT encoder and decoder might be better than the same amount of POS, NE or syntactic examples. If lots of data are available, then the network could learn the morphology or syntax better than from supervised training. For future work, it might be beneficial to explore the limits, for which data sizes the supervision in linguistic tasks still helps.

The authors of these studies also used an attentional sequence-to-sequence model, which is replaced by Transformer nowadays. It is desirable to adapt their approaches to Transformer. The results of Kiperwasser and Ballesteros [2018] on standard dataset and the novel deep model by Zaremoodi and Haffari [2018] seem to be promising and valuable for future research.

2.2 Enriching Target through Multi-Task NMT

Recently, models were proposed that are able to simultaneously translate and parse the target [Eriguchi et al., 2017], or predict target side CCG supertags (from which constituency parse trees can be retrieved) and translate [Nadejde et al., 2017].

The authors of both papers adapt the attentional sequence-to-sequence model. Eriguchi et al. [2017] propose a new model combining NMT with recurrent neural network grammars for joint translation and parsing. They report improvements in low-resource conditions, with 100k training sentences.

2.2.1 Interleaving vs Multi-Decoder in sequential model

Nadejde et al. [2017], on the other hand, use the standard encoder and decoder. The supertags are interleaved into the target sequence. Nadejde et al. [2017] report improvement on one high-resource task, if syntactic information is provided in source word embeddings as well. They also show an evidence that the interleaving approach outperforms multi-tasking, a model with two separate decoders, one for MT and one for supertags. See the illustration in Figure 2.3.

The interleaving approach was also successfully used by Tamchyna et al. [2017] for modelling target side morphological features.
2.2.2 Interleaving in Transformer

Tomáš Musil (reported in Appendix B6 in Bojar et al. [2018]) experiments with interleaving in Transformer model to enrich the target with tectogrammatical (semantic), analytical (syntactic) and morphological features from PDT Bejček et al. [2011]. He used a sample of 1M sentences from CzEng [Bojar et al., 2016] for English-to-Czech translations. His experiments do not show improvement over the baseline, the best experiments are actually 2 BLEU points worse and did not succeed in manual evaluation. Comparable results were achieved by Ondřej Bojar also with the sequential model (same report).

Discussion

Tomáš Musil claims that in translation into Czech, a language with rich morphology, the errors in morphology surpass the benefits of extra linguistic knowledge. He also notices that the model enriched with semantic parts-of-speech handles punctuation better, but the overall quality is worse.

Another reason, why interleaving with tectogrammatics and another features did not help, is because it was trained on automatically annotated dataset, which may not be accurate. Especially the part of PDT manually annotated on t-layer, on which the automatic annotation tool was trained, is relatively small. It has 39k sentences, 652k tokens.\footnote{https://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/en/html/ch03.html#a-data-full}

The last reason is that the attentative model is substantially different from the sequential. The model handles the words rather as bags of words than as sequences. The relative positions of neighboring words are not as remarkable as in the sequential model, and it requires far more training to handle them properly. Since comparable experiments with sequential model showed similarly bad results, we suppose the non-sequentiality of Transformer was only one of the unsolved problems.
2.3 Enriching Source by Linguistic Features in Word Embeddings

The main topic of our thesis is enriching NMT with multitasking. To put it into context of other approaches, we describe one alternative way for using linguistic resources in NMT: linguistic input features in word embeddings.

Sennrich and Haddow [2016] describe enriching encoder input of attentional encoder-decoder NMT model [Bahdanau et al., 2014] with linguistic features. Bahdanau's standard encoder takes a word embedding vector on input. The authors replace it with the concatenation of embedding vectors trained on distinct linguistic features: lemmas, subword tags (representing the decomposition of words into subwords by marking the beginning, end or inside subword of a word), POS, dependency labels and morphological features (case, number, gender etc.). See Figure 2.4 for example.

![Figure 2.4](image)

Figure 2.4: Original dependency tree for the sentence “Leonidas begged in the arena .”, and its feature representation after BPE segmentation. Figure and caption reprinted from Sennrich and Haddow [2016].

The authors provide experiments on data from WMT16 shared translation task on German to English and English to German. Their training corpus consist of 4.2M sentence pairs with maximum length 50. They also evaluate linguistic features in low-resource setting: English to Romanian translation with 0.6M parallel training sentences. Furthermore, the authors used monolingual corpora for backtranslation and compared the results with state-of-the-art systems. They report significant improvements over the baseline and also over state of the art.
3. Basic Settings for Experiments

In this chapter, we describe and give reasons for the basic settings used in our further experiments. Firstly, we describe our dataset for German-to-Czech MT, then the NMT frameworks Tensor2Tensor and OpenNMT, a method for subword segmentation and vocabulary construction, and a method for automatic evaluation of candidate translations.

3.1 German-to-Czech Dataset

A secondary scope of our work is to explore current state-of-the-art in MT for German-to-Czech language pair. In this section we describe the dataset, which we further use for our experiments, its origin, preprocessing and cleanup.

3.1.1 Test and Validation

WMT shared translation task provided German-to-Czech test dataset in 2011 and 2013.\(^1\) They contain 3k parallel sentences from news domain. We use them for evaluation of our experiments. We use newstest2011 as validation dataset and newstest2013 as testset.

3.1.2 Training

The latest works focusing particularly to German-to-Czech MT used phrase-based approach [Bojar and Zeman, 2014, Tlustý, 2016]. Their authors used the Europarl corpus [Koehn, 2005] for training, which provides only 500k German-Czech sentence pairs. This amount is useful for PBMT, but for NMT, as we showed in Section 1.1.1, we need more data. We decided to use all publicly available parallel German-Czech that exist: Europarl v7 [Koehn, 2005] and OpenSubtitles2016 [Tiedemann, 2009]. The latter one is considered as a big, but low-quality resource. It consists of parallel subtitles of movies or series. The translations may be inaccurate, provided by amateurs and not by experts, and contain lots of noise. However, we do not have any other option anyway, so we decided to attempt to clean them and then use them as training corpora.

Preprocessing

We use the same preprocessing pipeline, which Dan Zeman used for his system “cu-zeman”, reported in Bojar and Zeman [2014]. It is the most recent and best system for German-to-Czech MT so far, which is available to the research community together with description of its internal structure and translation of a testset for comparison. All comparable systems evaluated on newstest2013 are uploaded to Euromatrix;\(^2\) “cu-zeman” achieved the highest BLEU-cased score, 14.2.

\(^1\)http://www.statmt.org/wmt13/translation-task.html
\(^2\)http://matrix.statmt.org/matrix/systems_list/1715
The preprocessing pipeline consists of tokenization and character normalization using the tools from Moses project [Koehn et al., 2007]. We do not apply truecasing.

**Europarl Cleanup**

In Europarl, there were sentences marked as Czech or German, which were actually in different languages. See example on Figure 3.1. We used a language detection tool `langdetect`\(^3\) to filter them out.

**OpenSubtitles Cleanup**

In OpenSubtitles2016, some sentences, originally the movie captions, are incorrectly aligned; the two consecutive captions are merged into one in only one of the parallel variants. We attempted to exclude such occurrences. We filtered out the sentence pairs, whose length of source and target differ in more than 16 words. We also excluded the sentences with less than 4 words.

Furthermore, we noticed that in many Czech sentences, words like *skvělé* (“amazing”) appear, in which a letter *l* (lowercase L) is substituted by *I* (uppercase I). In many fonts, this substitution is not recognizable, so it does not affect the quality of subtitles, but it is undesirable in MT. We attempted to substitute it back by regular expressions and simple rules, but we did not cover all occurrences.

This cleanup process is obviously suboptimal and can be further improved, but that is beyond a scope of this work.

Table 3.1 summarize preprocessing cleanup progress and sizes of used corpora. Table 3.2 shows final training dataset statistics.

---

\(^3\)[https://github.com/Mimino666/langdetect](https://github.com/Mimino666/langdetect)

---

Figure 3.1: Examples of low quality parallel sentences in Europarl.
Table 3.1: Training corpora sizes before and after cleanup

<table>
<thead>
<tr>
<th>dataset</th>
<th>parallel sent.</th>
</tr>
</thead>
<tbody>
<tr>
<td>original Europarl</td>
<td>555,399</td>
</tr>
<tr>
<td>cleaned Europarl</td>
<td>555,022</td>
</tr>
<tr>
<td>original OpenSubtitles</td>
<td>9,344,614</td>
</tr>
<tr>
<td>cleaned OpenSubtitles</td>
<td>8,212,089</td>
</tr>
<tr>
<td>cleaned Europarl+OpenSubtitles</td>
<td>8,767,111</td>
</tr>
</tbody>
</table>

Table 3.2: Final training data statistics.

<table>
<thead>
<tr>
<th></th>
<th>German</th>
<th>Czech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel sentences</td>
<td>8.8M</td>
<td></td>
</tr>
<tr>
<td>Tokens</td>
<td>89M</td>
<td>78M</td>
</tr>
<tr>
<td>Vocabulary size</td>
<td>807k</td>
<td>953k</td>
</tr>
</tbody>
</table>

### 3.2 NMT Frameworks

In this section, we describe the NMT frameworks used in our work, Tensor2Tensor and OpenNMT-py. We also provide the hyperparameter setting for the Transformer model, and we make notes about terminology specific for Tensor2Tensor.

#### 3.2.1 Tensor2Tensor

Tensor2Tensor\textsuperscript{4} [Vaswani et al., 2018] (T2T for short) is an open-source library of deep learning models and datasets. It is actively used and maintained by researchers and engineers within the Google Brain team\textsuperscript{5} and a community of users. It is well-suited for neural machine translation and includes the reference implementation of the state-of-the-art Transformer NMT model [Vaswani et al., 2017]. T2T is based on TensorFlow\textsuperscript{6} backend.

Our experience and comparisons with other frameworks showed T2T as robust and reliable for extensive NMT experiments. It is also perhaps the best choice to obtain state-of-the-art performance, thanks to the Transformer and the fact it was firstly implemented in T2T and it has been used and improved by a big community of users and developers. Vaswani et al. [2018] report higher BLEU score achievements on English-to-German and English-to-French translation tasks than current state-of-the-art models, with a fraction of their training cost. Similar achievement is reported by Popel and Bojar [2018] on Czech-to-English.

A disadvantage of T2T is that it is not easy (at least for not very experienced programmers) to modify the codebase of T2T to implement new features or use it as a library. The code of T2T is very complex, it is under rapid continuous development, and the documentation is not very elaborated.

\textsuperscript{4}https://github.com/tensorflow/tensor2tensor
\textsuperscript{5}https://ai.google/research/teams/brain
\textsuperscript{6}https://www.tensorflow.org/
3.2.2 OpenNMT

OpenNMT\textsuperscript{7} [Klein et al., 2017] is an open-source initiative for neural machine translation and neural sequence modeling. It was originally developed by Yoon Kim\textsuperscript{8} and Harvard NLP\textsuperscript{9}. A major source contribution and support come from SYSTRAN\textsuperscript{10} company.

OpenNMT has currently three main implementation branches:

- OpenNMT-lua (a.k.a OpenNMT) – the main branch, developed with Lua-Torch\textsuperscript{11}, a scientific computing framework with GPU support. It is stable, robust, well documented and suitable for use in production without any code modifications.

- OpenNMT-py – an OpenNMT-lua clone using PyTorch\textsuperscript{12}. Initially created by Facebook AI research team. This implementation is easy to extend by new features, its backend is kept simple. Therefore it is well-suited for research. It has Transformer model implementation and currently it doesn’t offer training on multiple GPUs.

- OpenNMT-tf – a TensorFlow alternative. It currently does not offer the Transformer model.

3.2.3 Training Tips for the Transformer Model

Popel and Bojar [2018] published a paper about their NMT experiments using Tensor2Tensor with Transformer model. They confirm the general mantra “more data and larger models” are necessary for better performance, although the size of data and models is always restricted. The authors provide practical tips for training regarding hyperparameter sets, batch size, learning rate, warmup steps etc.

Since exhausting hyperparameter tuning for Transformer training is demanding and it’s beyond a scope of this work, we decided to follow the recommendations of the authors. Therefore we use following hyperparameters for all our further experiments:

- \texttt{transformer\_big\_single\_gpu}\textsuperscript{13} model and hyperparameter set. It has 280M trainable parameters.

- batch size = 1500 – the authors claim the higher batch size, the better results. We use the same hardware as authors, batch size 1500 is on the edge of memory limits.

- warmup steps = 60k – with 30k, learning sometimes diverged.

\textsuperscript{7}http://opennmt.net/
\textsuperscript{8}http://yoon.io
\textsuperscript{9}http://nlp.seas.harvard.edu/
\textsuperscript{10}http://www.systransoft.com/
\textsuperscript{11}http://torch.ch/
\textsuperscript{12}https://pytorch.org/
\textsuperscript{13}See https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/models/transformer.py for details.
• vocabulary size = 100k shared vocabulary – the authors don’t describe the impact of vocabulary size, but we assume higher vocabulary size leads to lower out-of-vocabulary rate and allows model learn more. The size 100k is also on the edge of memory limits on our hardware.

The authors also trained a model for English-to-Czech translation in comparable conditions with WMT17 submissions. The Transformer model outperformed all of them after 8 days of training on 8 GPUs.

3.2.4 Note on Tensor2Tensor Terminology

In T2T, some terms are used differently then in standard deep learning theory. We explain them here. We follow Popel and Bojar [2018], Section 2.3.

• **Training Step** is one optimizer update, respectively the number of optimizer updates. This number also equals the number of (mini)batches that were processed.

Processing of each training step takes constant time on ideal hardware. Training steps can be converted to time by a simple linear function.

• **Batch Size.** In T2T, batch size is the number of training examples used by one GPU in one training step, including padding symbols. (In other frameworks and sequential models, batch size is usually specified as the number of sentence pairs.)

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. However, some padding is still needed, thus batch size only approximates the actual number of (non-padding) subwords in a batch.

• **Effective Batch Size** is the number of training examples consumed in one training step. The effective batch size is usually cca 90 % of batch size, 10 % are padding tokens. T2T logs them regularly. It must be multiplied by number of GPUs, because T2T defines batch size per one GPU. We use 1 GPU in all our experiments.

• **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. Since T2T reports only the number of training steps, we approximate the number epochs by multiplying the steps by the effective batch size and divide by the number of subwords in the training data.

In multi-task, we have specific data for each task. If not stated otherwise in our work, let *epoch* be over the union of the MT and additional task training data.

• **MT Epoch** is one complete pass over training data for the main MT task.
• **Number of Subwords.** T2T preprocesses the input by segmenting whole (and possibly unknown) words into subwords. The total number of subwords is given by the internal vocabulary size parameter (which is in fact only approximated, it may differ by 10% in reality).

High average number of subwords per word in the given dataset may correlate with low quality of segmentation.

### 3.3 Word Segmentation

The vocabulary size in T2T is restricted by practical limits given by GPU memory capacity, but the actual vocabulary of wordforms in natural language, and especially in highly inflected Czech or German, is very big. The state of the art technique to handle the rich morphology in NMT is to segment the words to subword units, so the overall vocabulary size of these units fits the practical limits.

The common methods for subword construction are byte pair encoding (BPE, Sennrich et al. [2016]), and SubwordTextEncoder, a method implemented in T2T toolkit. Both are linguistically uninformed, based on statistics. The most frequent wordforms in training data get a full vocabulary entry. The less frequent wordforms can be split to two or more subwords, by the frequency of their substrings. Rare or completely unknown words can be split up to single characters, the smallest subword units.

We follow the findings of Macháček et al. [2018]. Although the linguistically uninformed methods may be suboptimal, the roots can be split several ways and the network may not explicitly know about their relatedness, we conclude that the linguistically uninformed methods perform better, and also that SubwordTextEncoder outperformed BPE on the same German-to-Czech dataset as we use in this work.

In our experiments we use the same setting for data segmentation as in Macháček et al. [2018], the data are processed by SubwordTextEncoder trained on source and target side of training data. We selected shared vocabulary of size 100k, because we assume the big vocabulary enables the model to be more robust, it can learn more relations between subwords and translate better.

With vocabulary size 100k we leverage almost the full 11GB GPU memory. It showed that this vocabulary, together with batch size 1500, causes out-of-memory errors on GPU with 8GB memory.

### 3.4 Automatic Evaluation

We evaluate our candidate translations with automatic BLEU score [Papineni et al., 2002]. We report the cased variant computed by the bleu_wrapper implemented in T2T framework.\textsuperscript{14} The test and validation set was preprocessed with the same pipeline as the training data, which means character normalization and tokenization by tools from Moses toolkit [Koehn et al., 2007]. For simplicity, we compute the BLEU scores on the tokenized data.

\textsuperscript{14}https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/bleu_hook.py
We include the translations of the most promising experiments into the electronical attachment, so the reader can analyze and evaluate them with any tool or method on his own.
4. Multi-Tasking through Simple Alternating

In Chapter 2 we analyzed works related to multi-tasking for NMT. We see that multi-tasking with shared encoder and decoder by alternating the tasks in training can lead to better translations. In this chapter we explore a similar approach with the standard German-to-Czech translation task (high resource, 8.7M sentences) and Transformer model implemented in Tensor2Tensor.

4.1 Variable Parameters of Experiments

The multi-tasking model with simple alternating can be parameterized by many different ways. We do not enumerate an exhaustive list of all possible or promising parameters and values, but only the ones which we used in our experiments to analyze their impact to translation quality. The fixed parameters of our experiments (Transformer model implemented in Tensor2Tensor toolkit version 1.2.9, transformer_big_single_gpu hyperparameter set, batch size 1500, standard size German-to-Czech dataset, vocabulary of size 100k created by the T2T's default SubwordTextEncoder, etc.) were already described and justified in Chapter 3.

In parentheses, we provide abbreviations of the parameter and value which we will use further for brevity. We denote some parameters as “default” because many of our experiments share this setting and it would be inefficient to repeat it over and over. The reader should however note that it is the default only within our work.

4.1.1 Multitasking Model Settings

In this section we describe the multitasking model architectures and method for task identification.

Multitasking Architectures

- shared encoder and decoder (default) – We use the basic implementation of Tensor2Tensor framework, which is originally designed for single-task MT, without any modifications. Detailed hyperparameters for T2T are mentioned in Section 3.2.3.

- shared encoder, task-specific decoders (spec dec) – We adapted OpenNMT-py framework to support multi-decoder models.

Task Identification (TI)

Task identification is necessary for alternating multi-task MT with shared model.

- mark tasks (default) – We provide the task determination by adding a special token as the last symbol of the source sentence. The tokens are
Translate and Other. They start with Translate because this sequence of characters never appears in the training data, so it gives the token a unique meaning. We chose the letters Z for technical reasons, they are alphabetical symbols and SubwordTextEncoder does not split them implicitly, as it does on boundaries of alphabetical and non-alphabetical symbols. The second symbol is used for all secondary tasks.

- no marker – For baseline single-task MT, the task determination is not necessary.

4.1.2 Multitasking Variants

The following variants are applicable to the multi-tasking model with shared encoder and decoder.

- alternating (default) – We merge the parallel training data for all tasks into one single dataset. The source sides are marked with a task identification token, from which the model determines the intended task and the corresponding target.

- interleaving (interleaving) – Interleaving is an operation for merging two sequences into one. If we have sequences \( a_1, a_2, a_3, \ldots, a_n \) and \( b_1, b_2, b_3, \ldots, b_n \), we can interleave them to create \( a_1, b_1, a_2, b_2, a_3, b_3, \ldots, a_n, b_n \). In multitasking, we usually interleave tokens and their corresponding tags. We can also interleave two secondary tasks into one.

- enriching input (enrich) – Enriching input is interleaving MT source with additional linguistic information. For example, each named entity can be followed by a special marking token, or each word with its lemma.

See examples of multitasking variants in Figure 4.1.

4.1.3 Data Settings

In this section, the dataset settings are described.

Origin of Secondary Task Data

- automatically annotated training data (default) – Each source sentence from the training data is used twice, for MT (in a pair with the parallel target sentence) and for the secondary task, for which it is paired with its automatic annotation. It is not necessary to use the same training data for both tasks, but we do it and consider it as default.

- gold, manually annotated corpus (gold)
### Machine Translation (MT)

Ich nehme dich mit .  \(\rightarrow\) Svezu tě .
Ich habe immer das gleiche Gefühl .  \(\rightarrow\) Vždycky mám ten stejný pocit .

### Dependency tagging (DT) of source

Ich nehme dich mit .  \(\rightarrow\) nsubj root obj compound:prt punct
Ich habe immer das gleiche Gefühl .  \(\rightarrow\) nsubj root advmod det amod obj punct

---

### Alternating MT+DT of source

Ich nehme dich mit .  \(\_\_\text{Translate}\_\_\)  \(\rightarrow\) Svezu tě .
Ich nehme dich mit .  \(\_\_\text{DepTag}\_\_\)  \(\rightarrow\) nsubj root obj compound:prt punct
Ich habe immer das gleiche Gefühl .  \(\_\_\text{Translate}\_\_\)  \(\rightarrow\) Vždycky mám ten pocit .
Ich habe immer das gleiche Gefühl .  \(\_\_\text{DepTag}\_\_\)  \(\rightarrow\) nsubj root advmod det amod obj punct

### Interleaving MT+DT of target

Ich nehme dich mit .  \(\rightarrow\) Svezu root tě nmod . punct
Ich habe immer das gleiche Gefühl .  \(\rightarrow\) Vždycky obj mám nmod ten root stejný amod pocit nsubj . punct

### Enriching input

Ich nsubj nehme root dich obj mit compound:prt . punct
Ich nsubj habe root immer advmod das det gleiche amod Gefühl nsubj . punct
Ich nsubj habe root immer advmod das det gleiche amod Gefühl nsubj . punct

Vždycky mám ten pocit .

---

Figure 4.1: Example of multitasking variants: alternating, interleaving and enriching input.
Sampling of Training Data

In alternating setup, it is necessary to set the ratio of MT and secondary task training data. This is an equivalent technique to the “ConstantScheduler” suggested by Kiperwasser and Ballesteros [2018], which we mentioned in Section 2.1.3.

- 1:1 (default) – If we use this option together with automatically each training sentence pair is included once for MT and once for the
- 2:1 (2:1samp)
- no oversampling (no sampling) – We combine two corpora for MT and secondary tasks without respect to their relative sizes. In one of our experiments, this leads to 171:1 sampling.

Amount of Training Data

We experiment with two data setups for German-to-Czech translation. The dataset is described more in Section 3.1.

- standard (default) – All 8.7M training sentences.
- low resource (datasize:500k) – 500k training sentence pairs, selected randomly.

4.1.4 Secondary Tasks

We denote the MT task as the main task, and we train it jointly with one of the secondary tasks. The shared multi-task NMT model must dedicate some internal capacity for determining and switching the tasks, which could be used for translation otherwise. This is a multi-tasking cost and we assume it is bigger with more secondary tasks. Therefore we start our experiments with only one secondary task, either linguistic, or synthetic dummy task for reference.

Linguistic Tasks

- POS tagging – we use TreeTagger for German Schmid [1995]\(^1\) in two versions:
  - full (POSfull) – The original tagset with 54 tags.
  - small (POSSmall) – We restricted the number of tags to 24 by dropping all but first two symbols of each tag and labeling all punctuation symbols with one tag. This merges related POS categories to bigger ones, e.g. conjunctions are further represented by one tag, not by four.

\(^1\)See the tag table at http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-table.html.
• unlabeled dependency parsing (Dheads) – We use Universal Dependencies 2.0 (UD, Nivre et al. [2017]). On the target side for this task, source words are replaced by forms of their fathers in the dependency tree and the root is represented with a special token. We neglect the fact that word forms can be used multiple times in a sentence, so the recovered parse can be ambiguous. We assume this situation is very rare because in UD, inner nodes are content words and they are rarely repeated in one sentence.

• dependency tagging (Dtag) – Tagging with labels of edges in the UD dependency tree.

• dependency parsing and tagging (Dht) – Interleaving of the above mentioned tasks.

• named entities (NE) – We use two kinds of corpora for NE, gold and automatically annotated training data.

  – gold: We use manually annotated NoSta-D Named Entity Annotation corpus [Benikova et al., 2014] for German. This NE corpus consists of 519k tokens, 31k sentences from Wikipedia and online news. We trained it together with the parallel corpus for MT, 89M source tokens, 8.7M sentences. The ratio of MT and NE data is 171:1. The named entities are annotated in two layers. The first is the outer layer, e.g. an organization name, and the second layer is the nested named entity inside of a multi-word named entity, e.g. a city name inside of organization name. Inside-outside-beginning (IOB) tagging format is used, and 4 classes of named entities (person, location, organization, and other). We linearized the layers, classes and IOB format into a single tagset with 58 tags. We further replaced non-alphabetical characters from the tagset with alphabetical ones to disable SubwordTextEncoder’s implicit split of boundaries of alphabetical and non-alphabetical symbols.

  – auto: Stanford Named Entity Recognizer [Finkel et al., 2005] for automatic NE tagging. It uses simplified single-layer annotation without IOB tags, with 4 classes only (person, organization, location, miscellaneous).

We used them in following variants:

  – NE58 gold – NE tagging from gold dataset with all 58 above mentioned tags
  – NE2 gold – NE tagging with only 2 tags, “NE” and “other”
  – NE5 auto – Automatic NE tagging with 5 tags: person, organization, location, miscellaneous, and other (no NE).
  – NE2 auto – simplified automatic tagging with only two classes, “NE” and “other”

See Figure 4.2 for illustration of NE variants and Figure 4.3 for other linguistic tasks.
<table>
<thead>
<tr>
<th>source wordform</th>
<th>NoSta-D orig. layer 1</th>
<th>layer 2</th>
<th>NE58 gold 2 layers, 1 tagset</th>
<th>NE5</th>
<th>NE2 gold/auto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zudem</td>
<td>O</td>
<td>O</td>
<td>OxO</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>veröffentlichte</td>
<td>O</td>
<td>O</td>
<td>OxO</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>er</td>
<td>O</td>
<td>O</td>
<td>OxO</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>zwei</td>
<td>O</td>
<td>O</td>
<td>OxO</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Songbücher</td>
<td>O</td>
<td>O</td>
<td>OxO</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>,</td>
<td>O</td>
<td>O</td>
<td>OxO</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Songs</td>
<td>B-OTH</td>
<td>O</td>
<td>BuOTHxO</td>
<td>MISC</td>
<td>NE</td>
</tr>
<tr>
<td>From</td>
<td>I-OTH</td>
<td>O</td>
<td>InOTHxO</td>
<td>MISC</td>
<td>NE</td>
</tr>
<tr>
<td>The</td>
<td>I-OTH</td>
<td>O</td>
<td>InOTHxO</td>
<td>MISC</td>
<td>NE</td>
</tr>
<tr>
<td>Hills</td>
<td>I-OTH</td>
<td>O</td>
<td>InOTHxO</td>
<td>MISC</td>
<td>NE</td>
</tr>
<tr>
<td>Of</td>
<td>I-OTH</td>
<td>O</td>
<td>InOTHxO</td>
<td>MISC</td>
<td>NE</td>
</tr>
<tr>
<td>Tennessee</td>
<td>I-OTH B-LOC</td>
<td>InOTHxBuLOC</td>
<td>LOCATION</td>
<td>NE</td>
<td>NE</td>
</tr>
<tr>
<td>und</td>
<td>O</td>
<td>O</td>
<td>OxO</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Arthur</td>
<td>B-OTH B-PER</td>
<td>BuOTHxBuPER</td>
<td>PERSON</td>
<td>NE</td>
<td>NE</td>
</tr>
<tr>
<td>Smith</td>
<td>I-OTH I-PER</td>
<td>InOTHxIuPER</td>
<td>PERSON</td>
<td>NE</td>
<td>NE</td>
</tr>
<tr>
<td>'</td>
<td>I-OTH I-PER</td>
<td>InOTHxIuPER</td>
<td>PERSON</td>
<td>NE</td>
<td>NE</td>
</tr>
<tr>
<td>s</td>
<td>I-OTH I-PER</td>
<td>InOTHxIuPER</td>
<td>PERSON</td>
<td>NE</td>
<td>NE</td>
</tr>
<tr>
<td>Original</td>
<td>I-OTH</td>
<td>O</td>
<td>InOTHxO</td>
<td>MISC</td>
<td>NE</td>
</tr>
<tr>
<td>Song</td>
<td>I-OTH</td>
<td>O</td>
<td>InOTHxO</td>
<td>MISC</td>
<td>NE</td>
</tr>
<tr>
<td>Folio</td>
<td>I-OTH</td>
<td>O</td>
<td>InOTHxO</td>
<td>MISC</td>
<td>NE</td>
</tr>
<tr>
<td>No.1</td>
<td>I-OTH</td>
<td>O</td>
<td>InOTHxO</td>
<td>MISC</td>
<td>NE</td>
</tr>
<tr>
<td>.</td>
<td>O</td>
<td>O</td>
<td>OxO</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Figure 4.2: Example of variants of named entity annotation of German source sentence “Besides, he published two songbooks, Songs From The Hills Of Tennessee and Arthur Smiths’s Original Song Folio No.1.” (NE columns) and the original NoSta-D 2 layer annotation, which we transformed into NE58 gold.
Translation (MT)
Ich nehme dich mit.  →  Svezu tě.
Ich habe immer das gleiche Gefühl.  →  Vždycky mám ten stejný pocit.

TreeTagger POS (POSfull)
Ich nehme dich mit.  →  PPER VVFIN PPER PTKVZ $.
Ich habe immer das gleiche Gefühl.  →  PPER VVFIN ADV ART ADJA $.

Universal Dependency Trees

Dependency Parsing (Dheads)
Ich nehme dich mit.  →  nehme root nehme nehme nehme
Ich habe immer das gleiche Gefühl.  →  habe root Gefühl Gefühl Gefühl
habe habe

Dependency Tagging (Dtags)
Ich nehme dich mit.  →  nsubj root obj compound:prt punct
Ich habe immer das gleiche Gefühl.  →  nsubj root advmod det amod obj punct

Dependency Parsing and Tagging (Dht)
Ich nehme dich mit.  →  nehme nsubj root root nehme obj
nehme compound:prt nehme punct
Ich habe immer das gleiche Gefühl.  →  habe nsubj root root Gefühl
advmod Gefühl det Gefühl amod habe
obj habe punct

Named Entity Recognition (NE5 auto)
Ich bin Petr.  →  O O PERSON O
Ich fahre nach Prag.  →  O O O LOCATION O

Figure 4.3: Example of linguistic additional tasks.
Synthetic Dummy Tasks

We use the synthetic dummy secondary tasks to empirically measure the actual benefits of linguistically adequate tasks to MT. We give following reasons for it:

Let \( P \) be the difference of MT quality of the baseline and multi-tasking model for MT and one secondary task. Thus, \( P \) is positive, if the multi-tasking model outperformed baseline, and negative otherwise. We assume that \( P \) consists of following components: multi-tasking cost \( M \) (due to necessity of task identification), the cost for performing the secondary task \( S \), and \( B \), the benefit of the secondary task to MT. We further assume that \( P = B - S - M \).

Let us consider a model \( d \) with a dummy secondary task and an identical model \( l \) with a linguistic secondary task, such that the linguistic task is more difficult than \( d \), so we can assume that \( S_l > S_d \). We train and evaluate both models and empirically measure \( P_l \) and \( P_d \).

If \( P_l > P_d \), then \( B_l - S_l - M_l > B_d - S_d - M_d \). In ideal conditions, the multi-tasking cost is constant, so that \( M_l = M_d \), and \( B_d = 0 \), because the dummy task is unrelated to MT. We get \( B_l - S_l > -S_d \) and further \( B_l > S_l - S_d > 0 \). This is a proof that if \( l \) outperforms \( d \) on MT task, and our assumptions about the costs and benefit are realistic, than the benefit of the linguistic secondary task \( l \) to MT is positive. We can say that even if the benefit is not big enough to overcome the multi-tasking cost and beat the baseline.

From this reason we prepare and evaluate the dummy tasks, which are not related to MT and are simpler than the linguistic tasks, which we want to evaluate. We use following dummy secondary tasks:

- **enum subwords** – All subwords in source are replaced by a dummy token (ˇˇˇ). We assume this task is very easy for attentative architecture.
- **count subwords** – The number of source subwords is emitted as a decimal number, e.g. 11.
- **enum words** – The full words are replaced by dummy token ˇˇˇ. We assume that if the model answers correctly, then it is able to detect word boundaries. This is evidently simpler than tagging.
- **count number of words in a source sentence** (count words)
- **repeat source** (repeat) – The source tokens are repeated. This can be used for measuring benefits of the Dheads task, which also has to emit parts of source.

See examples on Figure 4.4.
Figure 4.4: Example of synthetic dummy tasks.
4.1.5 Designing Vocabulary

The open vocabulary of natural languages must be compressed into a relatively small fixed size vocabulary for the NMT model. In simple alternating or interleaving multitasking, the vocabulary must cover all sides of all tasks. Since we consider MT as the main task, and the secondary tasks as auxiliary, we suggest several strategies for combining the task-specific vocabularies, which we further evaluate.

Table 4.1 shows individual vocabulary sizes of different tasks. Predicting heads in dependency trees shares the vocabulary with MT source (only the words not appearing as inner nodes are missing), while in POS tagging, dependency tagging and NER only a small fixed number of tags appears in the target.

<table>
<thead>
<tr>
<th>task</th>
<th>types</th>
<th>tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT target</td>
<td>952k</td>
<td>78M</td>
</tr>
<tr>
<td>MT source</td>
<td>807k</td>
<td>89M</td>
</tr>
<tr>
<td>count words</td>
<td>190</td>
<td></td>
</tr>
<tr>
<td>Enum Words/Subwords</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>repeat</td>
<td>807k</td>
<td></td>
</tr>
<tr>
<td>Dheads</td>
<td>670k</td>
<td></td>
</tr>
<tr>
<td>Dtag</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>POSfull</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>POSsmall</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>NE</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Size of training MT vocabulary before segmentation and size of target side vocabularies of additional tasks.

The suggested strategies vary in the following parameters:

Input for SubwordTextEncoder

SubwordTextEncoder is a module of T2T toolkit. It creates a vocabulary of subword units from dataset provided on its input (which is, in the default setting for single-task MT, identical to the training data), and then it segments the raw text into these units. In our multitasking experiments, MT is the main task and secondary tasks are auxiliary. We experiment with emphasizing either MT, or neither task in vocabulary design.

- balanced mix (default) – Concatenate source and target sides for both MT and additional task training data. Consequently run SubwordTextEncoder on it.

  Advantage: STE may find optimal segmentation for given combination of tasks.

  Disadvantage: Imagine e.g. Dheads task: The ratio of words from source and target languages in the concatenated dataset is 3:1. Source language appears in MT source, additional task source and additional task target. All sides are equal in length. On the other hand, the target language appears only on MT target side. Then the distribution of most frequent subwords
promotes source language and may result in good segmentation of source and splitting target to many short subwords for which the neural network needs more processing steps to cover the meaning.

- Train on MT (voc:MT) – train STE only on MT source and target. The tags from secondary task target are either appended to the finished vocabulary (denoted as +tags:yes), or segmented as normal (unseen) words using the MT motivated STE (denoted as +tags:no).

  Advantage: This approach might be good for tagging. The tags do not corrupt the subword distribution of natural language and obviously the model will cope them as standalone words.

  Disadvantage: For example for Dheads, the words appearing often as inner nodes of dependency trees may be neglected in statistical distribution for most frequent subwords, and then the additional task target may be segmented suboptimally.

**Share Dictionary Between Tasks**

It might be beneficial to ensure the tags have unique meaning, separate entries in vocabulary and make sure that they are represented as single subwords.

Therefore, the strategies for sharing dictionary between tasks are:

- distinct (default) – the tags appearing only in additional task target start with ˇZˇZˇZ. We also make sure they do not contain non-alphabetic characters, because STE would split them implicitly.

- shared (vocshr)

### 4.2 Experiments with Simple Alternating Setup

In this section, we provide results of our experiments with simple alternating setup. We analyze suitable stopping criterion, the benefits of linguistic secondary task, experiments with small training dataset, design vocabulary decisions, adequacy of linguistic knowledge for MT, finetuning the models to MT, impact of different task sampling, and exploiting named entities.

#### 4.2.1 Stopping Criterion

The training of deep learning models is usually stopped when the model is considered as overtrained. It is a point where validation loss starts increasing and validation quality starts decreasing. The final model is usually taken at the point with maximal validation quality before overfitting.

In NMT, model overfitting is rare when training on big datasets and big models. In such cases, we usually stop training at some point reachable in affordable time where validation quality stops increasing fast, although higher scores could still be reached with more computation time. Such a point is usually only vaguely defined from simple observation of the validation learning curves. This technique is obviously suboptimal, but simple.
Figure 4.5: Validation BLEU on MT task over training steps on best runs for each secondary task. Baseline is single-task MT.

We use the BLEU score [Papineni et al., 2002] as the quality metric for validation despite its well-known drawbacks. We have only one reference translation for our validation set. As native speakers of the target language, we see that BLEU score gives low rating to translations which are already correct, but not close enough to the reference. The best BLEU score of our baseline model is around 18 and it does not grow much higher. Manual rating or better automatic metrics is desired, but it is difficult and beyond the scope of this work.

Figure 4.5 shows validation BLEU over training steps for the baseline and best runs for each secondary task. Due to time constraints we ended the training of many runs (but not of all) at 1M steps. In the following, we report the quality scores at 600k training steps. From the perspective of Figure 4.5, it seems the learning is almost converged at 600k steps, at least enough to observe comparable and significant differences between runs.

4.2.2 Simple Multi-Tasking with Linguistic Resources

As Niehues and Cho [2017] confirmed (recall Chapter 2), multitasking with linguistic resources can outperform baseline with simple alternating approach. Our hypothesis is that this may work also with Transformer model.

Table 4.2 summarizes all experimental runs together with their validation BLEU scores at 600k steps of training (or at the nearest available checkpoint, if 600k sharp were not available). In this table, we report “line accuracy” for the secondary tasks, which is in our case the average of the percentage of correctly predicted words or tags in each line of evaluation set. If the predicted sentence is longer than the reference, we simply drop the remaining tokens. If the predicted sentence is shorter, then the missing tokens are counted as incorrect. Furthermore, we report the average number of subwords per one word. We believe it reflects the overall quality of the vocabulary, which in turn affects the overall translation performance.

Based on observations from Table 4.2 we selected best runs for each linguistic additional task, when trained with T2T, shared encoder and decoder, and our standard size dataset. From Figure 4.5, we see that none of multi-tasking methods
Table 4.2: Summary of all experimental runs using T2T at 600k training steps or the nearest available checkpoint. BLEU scores and average numbers of subwords are reported on validation data. Abbreviations of experimental runs are explained in Section 4.1. The symbols d/y/n/- denote “default”, “yes”, “no”, and “not applicable”, respectively.
outperformed the baseline in BLEU score under the same model size and training time conditions.

It is important to note that the multi-tasking runs spend half of the training time in learning the secondary task. Figure 4.6 shows validation MT BLEU over MT epochs. In other words, the x-axis here reflects the amount of processed MT data, not the training time. One epoch is approximately 80M tokens of parallel texts. From this perspective, we can see that multi-task learning helps the model learn to utilize the data faster than single-task MT, although the multi-tasking models finally converge to lower BLEU scores than the baseline.

Figure 4.6: Validation BLEU on MT task over MT epoch on best runs for each additional task. One epoch has 89M/78M tokens of source and target, respectively, or 8.7M parallel sentences. Baseline is single-task MT.

4.2.3 Experiments with Small Training Dataset

The experiments in previous sections were run on all publicly available parallel data for German-to-Czech translation. We provide experiments with a small randomly selected subset of the whole dataset to test the hypothesis that multi-tasking helps in low-resource conditions. The contrastive summary of both datasets follows in Table 4.3.

<table>
<thead>
<tr>
<th>dataset</th>
<th>parallel sentences</th>
<th>source types</th>
<th>tokens</th>
<th>target types</th>
<th>tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>big</td>
<td>8.8M</td>
<td>807k</td>
<td>89M</td>
<td>953k</td>
<td>78M</td>
</tr>
<tr>
<td>small</td>
<td>500k</td>
<td>160k</td>
<td>5.1M</td>
<td>222k</td>
<td>4.5M</td>
</tr>
</tbody>
</table>

Table 4.3: Contrastive summary of small and big training datasets for German-to-Czech translation.

Table 4.4 shows BLEU scores at 600k training steps and Figure 4.7 the overall MT validation BLEU curve over training steps. Joint double-task training on dependency tagging and MT outperformed single-task MT by 3 BLEU points on
small dataset, but the translation quality is still approximately 8 BLEU points worse than if we train on the big dataset. However, we can conclude that joint multi-task learning with dependency parsing is beneficial in low-resource conditions.

<table>
<thead>
<tr>
<th>run</th>
<th>vec:MT</th>
<th>no:tag</th>
<th>dev</th>
<th>dev</th>
<th>appx</th>
<th>avg # subwords:</th>
<th>MT</th>
<th>target</th>
<th>source</th>
<th>trans</th>
<th>ref</th>
<th>task 2</th>
<th>tgt</th>
<th>ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline, data size: 8.8M</td>
<td>d n n</td>
<td></td>
<td>17.90</td>
<td>-</td>
<td>605-409</td>
<td>8.75</td>
<td></td>
<td>1.20</td>
<td>1.24</td>
<td>1.10</td>
<td>1.20</td>
<td>1.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dtag, data size: 500k</td>
<td>d n n</td>
<td></td>
<td>10.13</td>
<td>81.55</td>
<td>600000</td>
<td>49.36</td>
<td></td>
<td>1.18</td>
<td>1.27</td>
<td>1.13</td>
<td>1.23</td>
<td>1.10</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>baseline + MT marker</td>
<td>d n n</td>
<td></td>
<td>7.26</td>
<td>-</td>
<td>600000</td>
<td>143.36</td>
<td></td>
<td>1.12</td>
<td>1.25</td>
<td>1.11</td>
<td>1.24</td>
<td>1.11</td>
<td>1.12</td>
<td>1.25</td>
</tr>
<tr>
<td>baseline, data size: 500k</td>
<td>d n n</td>
<td></td>
<td>6.83</td>
<td>-</td>
<td>600000</td>
<td>151.58</td>
<td></td>
<td>1.14</td>
<td>1.25</td>
<td>1.11</td>
<td>1.24</td>
<td>1.11</td>
<td>2.00</td>
<td>1.25</td>
</tr>
</tbody>
</table>

Table 4.4: Contrastive summary of experimental runs with the small and big dataset. BLEU scores and average numbers of subwords are reported on validation data.

Figure 4.7: Validation BLEU on MT task over training steps for small and big dataset. Baseline is single-task MT.
4.2.4 Designing Vocabulary for Multi-Task

We hypothesize that MT quality in multi-task setting may be positively affected by emphasizing MT task in vocabulary design. We also considered separate vocabularies for tags and MT, and ensuring that NMT model handles the tags as standalone words even after segmentation to subwords.

Table 4.5 and Figure 4.8 summarize experimental runs with different vocabulary design techniques described in Section 4.1.5. We conclude:

- **For tagging tasks** (Dtag, POSfull, POSsmall, NE), the basic setting with balanced mix of MT and secondary task data on SubwordTextEncoder input achieves better BLEU scores than emphasizing MT source and target only. Mixing the task vocabularies does not hurt the performance, probably because the overlap of subword types is not significant.

- **For unlabeled dependency parsing**, which is in our case predicting fathers in the dependency tree, holds the same observation as for tagging. Balanced mix for all sides of all tasks, which in fact leads to ratio 3:1 of MT source:target data on input for SubwordTextEncoder, achieves better MT BLEU score than vocabulary generated from MT data only. However, if we sample the MT and dependency parsing training data in ratio 2:1 (compare Dheads 1:2samp vs Dheads, voc:MT, +tags:no, 1:2samp in Figure 4.8e), the impact of vocabulary design seems to be insignificant.

- **For interleaved dependency parsing and tagging** (Dht), same observations hold.

- **For repeating source** as the secondary task, we observe the MT advantaging technique (voc:MT) seems to be better. We hypothesize that this is because repeating source does not require any linguistic knowledge, and therefore does not benefit from a lower average number of source and target subwords, while the linguistic tasks do.

- **For counting words**, we explain the 2 BLEU points loss achieved by mixed balanced and MT vocabulary by the fact that the MT data do not contain all decimal numbers for representation of the word counts. In the first 50k sentences of training data, there are 5 occurrences of numbers, which STE with the mixed balanced vocabulary represents with 2 subwords, but in STE with MT vocabulary there are 15 such occurrences. The secondary task was therefore more difficult, which is reflected also in 20% worse accuracy (see Figure 4.5). Since the tasks can be mixed within each batch, the low-quality predictions of the secondary task probably confuse the gradient away from optimal point for both tasks, and it results in lower quality of the MT task.
Figure 4.8: Validation BLEU over training steps for comparison of vocabulary design techniques on different kinds of linguistic and synthetic additional tasks.
Table 4.5: Summary of all experimental runs using T2T at 600k training steps or at nearest available checkpoint. BLEU scores and average numbers of subwords are reported on validation data. Abbreviations of experimental runs are explained in Section 4.1.
4.2.5 Adequacy of Linguistic Knowledge for MT

Linguistic knowledge (as POS tagging or parsing) is required by MT, and MT provides some kind of linguistic knowledge. When we train an NMT model jointly for an MT and a linguistic task, we hope that the internal hidden components of the neural network responsible for each task will overlap and benefit from the relatedness of MT and the respective task, so the overall model learns two tasks better than two separate models.

On the other hand, in shared encoder and decoder multi-tasking architecture, some hidden components of NN must be responsible for determining the tasks and providing task-specific knowledge. This obviously employs some capacity of the network which could be utilized for language understanding in single-tasking MT model. For this reason, the secondary task can also decrease performance.

We test the overall behavior of multi-tasking models utilizing POS tagging, dependency parsing and named entities in contrast with linguistically inadequate tasks: enumerating subwords and counting the number of subwords (which should be very easy for attentative architecture), enumerating words, counting words and repeating the source. These tasks are described in Section 4.1 in more details.

Table 4.6 and Figure 4.9 show the validation quality of the additional task processing. Corresponding MT BLEU is in Figure 4.10. The scores are reported against automatically annotated data, which may be inaccurate. However, we can still conclude the following:

- The double-tasking models learned all second tasks quite well, with accuracy from 75 to 99 %, which is better than the baseline classifier (predicting the most frequent class). Comparable single-tasking models for parsing and POS tagging with the same capacity, architecture and hyperparameters achieved horrible performance of 7% (see single-task Dtag and single-task POSfull in Table 4.2 at the bottom). We conclude that the double-tasking with MT helped the models to perform the linguistic tasks.

- MT with dependency tagging (Dtag), POS tagging (POSfull) and unlabeled dependency parsing (Dheads) as secondary tasks outperformed MT with linguistically inadequate tasks (repeat, counting words, enumerating words etc.). See Figure 4.10. Table 4.6 shows that MT+Dtag achieves 16.36 BLEU scores, while the baseline is 17.90. The latter two are less than 0.2 BLEU worse. We consider this as an evidence that these models contribute to better MT, however, some settings still decrease the overall performance.

- Task determination overhead and distinct output formats for MT and linguistic tasks may be the limiting factors preventing to outperform the baseline. We hypothesise that distinct decoders for each task may lead to overall improvement. On the other hand, Nadejde et al. [2017] shows the opposite, shared interleaving model for predicting target side CCG supertags outperformed multi-decoder model.

- The more fine-grained POS tagging into 54 tags contributes to MT 2.3 BLEU points more than the pruned tagset with 24 tags (compare POSfull and POSsmall in Table 4.6). Both models achieve comparable tagging accuracy around 97%.
We observe and discuss similar behavior with 58 vs 2 NE tags in Section 4.2.8.

<table>
<thead>
<tr>
<th>run</th>
<th>MT BLEU</th>
<th>prec.</th>
<th>rec.</th>
<th>F1</th>
<th>acc.</th>
<th>base.</th>
<th>appx. step</th>
<th>epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>17.90</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>605409</td>
<td>8.75</td>
</tr>
<tr>
<td>single-task MT+marker</td>
<td>16.53</td>
<td>81.87</td>
<td>81.73</td>
<td>81.10</td>
<td>88.58</td>
<td>14.28</td>
<td>6001309</td>
<td>8.18</td>
</tr>
<tr>
<td>Dtag, vocshr</td>
<td>16.52</td>
<td>97.22</td>
<td>97.19</td>
<td>97.10</td>
<td>97.89</td>
<td>21.68</td>
<td>6001322</td>
<td>4.15</td>
</tr>
<tr>
<td>POSfull, vocshr</td>
<td>16.40</td>
<td>75.38</td>
<td>75.13</td>
<td>73.67</td>
<td>77.56</td>
<td>5.94</td>
<td>6052982</td>
<td>4.13</td>
</tr>
<tr>
<td>Dheads</td>
<td>15.70</td>
<td>96.87</td>
<td>96.87</td>
<td>96.87</td>
<td>96.87</td>
<td>3.53</td>
<td>6014772</td>
<td>4.14</td>
</tr>
<tr>
<td>count words, vocshr</td>
<td>14.12</td>
<td>96.41</td>
<td>96.37</td>
<td>96.27</td>
<td>97.06</td>
<td>21.68</td>
<td>6076114</td>
<td>2.32</td>
</tr>
<tr>
<td>POSsmall, vocshr</td>
<td>13.89</td>
<td>99.90</td>
<td>99.90</td>
<td>99.90</td>
<td>99.90</td>
<td>3.36</td>
<td>6038284</td>
<td>1.65</td>
</tr>
<tr>
<td>enum subwords</td>
<td>12.16</td>
<td>96.12</td>
<td>96.17</td>
<td>96.14</td>
<td>99.83</td>
<td>-</td>
<td>6000000</td>
<td>2.62</td>
</tr>
</tbody>
</table>

Table 4.6: Validation quality metrics of secondary tasks. Top: Scores for classification, precision, recall, and F1 with macro averaging, accuracy and baseline (accuracy of predicting most common class). Bottom: mean squared error and mean absolute error for regression tasks (counting or enumerating correct count of words or subwords).

![Figure 4.9: Validation line accuracy of linguistic (bold lines) and dummy secondary tasks (thin lines) over training steps.](image)
Figure 4.10: Validation BLEU of MT+linguistic (bold lines) and dummy secondary tasks (thin lines) over training steps.

4.2.6 Finetuning of Multi-Task Model to MT

We trained double-task MT+Dheads model and tried to continue training with MT data only to adapt it for MT. Our hypothesis states that the joint model could capture linguistic knowledge better than baseline MT, and it could outperform the baseline after utilizing the full model capacity, including hidden components previously dedicated to Dheads task, for MT.

We started the finetuning of the Dheads model after 100k training steps, when the network was not yet fully trained to MT, and at 1M steps, when the training almost converged. Figure 4.11 shows the validation MT BLEU curve over training steps.

Figure 4.11: Validation MT BLEU over training steps for adaptation of MT+Dheads models to MT.

Neither of the models outperformed the baseline model without task determination token. Model finetuned from 100k showed the same performance as MT with task determination on source. Model finetuned from 1M steps increased performance to baseline level, but then the overall performance dropped.
4.2.7 Task Sampling

For our multi-tasking experiments, we chose the same amount of MT and secondary task training data, although this distribution may not be optimal. We assume that MT is more difficult than parsing, so the NMT model needs to process more MT training data. If we provide the NMT model more data for parsing than needed, then the MT quality may be corrupted by pushing the gradient away from MT optimum, while the model has already reached the optimal point for parsing and cannot learn anything new.

For contrasting experiments with different data sampling we used MT+Dheads task and identical vocabulary for each experimental run, to avoid any effects of vocabulary design. The vocabulary is created by SubwordTextEncoder from MT source and target data only. Special tokens for task determination and dependency tree root are appended to the finished vocabulary.

Table 4.7 and Figure 4.12 show the results. We see that the 2:1 sampling of MT vs Dheads training data leads to good parsing quality (actually 2 F1 points lower than 1:1 sampling) and to higher MT BLEU, which is approximately in the middle of the scale between MT single-task and 1:1 sampling. We hypothesize this is general behavior, but further investigation is needed.

<table>
<thead>
<tr>
<th>MT vs Dheads data sampling</th>
<th>MT BLEU</th>
<th>Dheads BLEU</th>
<th>prec.</th>
<th>recall</th>
<th>F1</th>
<th>appx step</th>
<th>appx epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:0 (MT only)</td>
<td>16.53</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>600000</td>
<td>8.18</td>
</tr>
<tr>
<td>2:1</td>
<td>15.71</td>
<td>62.59</td>
<td>75.53</td>
<td>75.29</td>
<td>73.87</td>
<td>600000</td>
<td>4.06</td>
</tr>
<tr>
<td>1:1</td>
<td>14.49</td>
<td>63.32</td>
<td>77.27</td>
<td>76.86</td>
<td>75.56</td>
<td>600000</td>
<td>4.06</td>
</tr>
</tbody>
</table>

Table 4.7: Validation MT BLEU and parsing quality for task sampling experiments.

Figure 4.12: Validation MT BLEU over training steps for experiments with sampling of MT and Dheads training data for double-tasking. All runs have identical vocabulary.
4.2.8 Exploiting Named Entities

We experimented with exploiting named entity recognition on source side of German-to-Czech translation. Recall the variable settings in Section 4.1.4.

Table 4.8 and Figure 4.13 show results of experiments with named entities. We observe the following:

- All but one double-tasking models outperformed baseline classifier in NER, as we can see in Table 4.8. We conclude the models learned to recognize some NE.

- Two runs, MT with NE5 auto enriching source and NE58 gold with no oversampling, achieved comparable BLEU score as the baseline. As Table ?? shows, baseline and NE5 enriched source have BLEU 17.90, NE58 gold 17.75. Their validation BLEU curve in Figure 4.13 overlap each other. Manual evaluation is necessary to test the actual differences. It is possible that the enriched models translate NE better, or they just learned to ignore the secondary task or interleaved tags and they translate the same way as baseline.

- At 600k training steps, all the reported experiments vary a lot in passes over the training data, as we see in the last column of Table 4.8. It is caused by different sizes of datasets. The baseline run uses 8.7M parallel sentences. Baseline with the task identification marker uses the same data, but they are augmented with the one token per sentence. MT+NE with no oversampling uses two corpora, 8.7M sentences for MT and 31k sentences for NE.

- The more fine-grained NE classification, the higher BLEU score achieved (compare the BLEU curves in Figure 4.13. The NE annotations with 5 and 58 classes showed better score than the same restricted to 2 classes. We suppose that for the 58 classes, a deeper linguistic knowledge is necessary and that more stacked layers of the encoder or decoder are employed and shared by the MT and NER tasks.

This behavior proposes potentially promising further experiments incorporating semantics. A small amount of high-quality data is probably more valuable than high amount of low-quality.
<table>
<thead>
<tr>
<th>run</th>
<th>sampling</th>
<th>MT BLEU</th>
<th>NE acc.</th>
<th>NE base.</th>
<th>step</th>
<th>epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td></td>
<td>17.90</td>
<td>-</td>
<td>-</td>
<td>605409</td>
<td>8.75</td>
</tr>
<tr>
<td>MT enrich source NE5 auto</td>
<td></td>
<td>17.90</td>
<td>-</td>
<td>-</td>
<td>605215</td>
<td>8.52</td>
</tr>
<tr>
<td>MT+NE58 gold</td>
<td>171:1</td>
<td>17.75</td>
<td>92.43</td>
<td>91.13</td>
<td>697687</td>
<td>11.38</td>
</tr>
<tr>
<td>MT+NER2 gold</td>
<td>171:1</td>
<td>16.62</td>
<td>87.04</td>
<td>91.13</td>
<td>600000</td>
<td>8.04</td>
</tr>
<tr>
<td>MT with TI marker</td>
<td></td>
<td>16.53</td>
<td>-</td>
<td>-</td>
<td>600000</td>
<td>8.18</td>
</tr>
<tr>
<td>MT+NE5 auto</td>
<td>1:1</td>
<td>14.12</td>
<td>98.00</td>
<td>93.89</td>
<td>605598</td>
<td>0.89</td>
</tr>
<tr>
<td>MT+NER2 auto</td>
<td>1:1</td>
<td>12.47</td>
<td>97.82</td>
<td>93.89</td>
<td>600000</td>
<td>2.62</td>
</tr>
</tbody>
</table>

Table 4.8: Validation MT BLEU and NE tagging quality for experiments with named entities. Accuracy (acc.) and baseline accuracy (base.) is reported. Note that for NE gold and automatic data, different datasets are used. Sampling parameter is the ratio of MT and secondary task training data.

Figure 4.13: Validation MT BLEU over training steps for experiments with sampling of MT and Dheads training data for double-tasking. All runs have an identical vocabulary.
4.3 Manual Evaluation

We provide manual evaluation of the most promising runs to compare them with the baseline. We ask and answer following the questions:

- Is the translation of the baseline model better at 250k steps, or at 1M steps?
  
  The validation BLEU curve of the baseline starts to flatten at 250k training steps, where the BLEU score is 17.03. At 1M steps, BLEU is 18.13, as we see e.g. in Figure 4.13. Maybe the translation is already good at 250k steps, and after this point, the model only overfits to the training data and the actual quality is lower at 1M steps, despite the difference of 1.1 BLEU points.

- Is the baseline translation better than alternating multi-tasking with NE? Which variant handles NE better?
  
  We compare the baseline with NE5 enriching source and MT+NE58 gold, sampling 171:1. These runs achieved around 0.1 higher BLEU score than the baseline, but this difference is so small that it may be insignificant. At 1M steps, baseline is 18.13, NE5 enrich source 18.28, MT+NE58 gold 18.21.

4.3.1 Baseline at 250k vs 1M Steps

To compare the translation quality at 250k and 1M training steps, we used a tool quickjudge\(^2\) with our own extension to select random 100 sentences from newstest2011, our validation dataset, translated by the two competing systems. We prepared a file for manual evaluation, where the German source, Czech reference and two candidate translations in random order with hidden origin were presented. The annotator had to mark one of the candidate translations with labels “substantially better” or “better”, or both candidates with labels “equally bad” or “equally good”. See example on Figure 4.14.

```
source  Uli Gsell hat sich auch welche besorgt , insgesamt etwa 400 Kilo .
ref     Uli Gsell si také nějaké pořídil , celkem asi 400 kilogramů .
sb      Uli Gsell si jich taky pár koupila . Celkem asi 400 kilo .
        Uli Gsell taky . Celkem 400 kilo .
```

Figure 4.14: An example of annotation of the sentence *Uli Gsell also got some of them. In total around 400 kilograms*. The first sentence is German source, the second is Czech referential translation. Two candidate translations follow, the first one is marked as “substantially better” and in fact is provided by the model at 250k steps.

In total, 78 sentences were judged. Table 4.9 summarizes the results. It shows that 15 sentences of the model at 1M steps were marked as “substantially better”, but only 6 such sentences belong to the other model. The numbers of “better” sentences are very close (9 versus 11), but if we count the number of “substantially better” or “better”, then the model at 1M steps wins, the score is

\(^2\)https://github.com/ufal/quickjudge
24:17. Therefore, we conclude the model at 1M steps is better than the model at 250k steps, and that in this case, the increasing validation BLEU is a sign of real quality improvement.

<table>
<thead>
<tr>
<th>model</th>
<th>subst. better</th>
<th>better</th>
<th>equally good</th>
<th>equally bad</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline 1M</td>
<td>15 (19%)</td>
<td>9 (12%)</td>
<td>10 (13%)</td>
<td>27 (35%)</td>
<td>18.13</td>
</tr>
<tr>
<td>baseline 250k</td>
<td>6 (8%)</td>
<td>11 (14%)</td>
<td></td>
<td></td>
<td>17.03</td>
</tr>
</tbody>
</table>

Table 4.9: Results of manual evaluation of the baseline run at 1M and 250k training steps on randomly selected 78 sentences from newstest2011. BLEU score is reported on the whole newstest2011.

### 4.3.2 Baseline vs NE58 Gold

To compare the baseline and NE58 gold with no oversampling, we prepared an annotation task the same way as in previous Section 4.3.1. In annotation we focused on two aspects, the overall quality of translation, and the quality of the translation of NE. Both aspects were marked the same way.

Table 4.10 summarizes the overall quality of translation. More than half of sentences, 56%, were either “equally good”, or “equally bad”. The ratio of “substantially better” or “better” sentences in baseline vs NE58 gold is 24:19. The baseline seems to be slightly better, but the difference is small. This evaluation is provided by only one annotator, others can have different opinion. Further evaluation is necessary to get more significant results.

<table>
<thead>
<tr>
<th>model</th>
<th>subst. better</th>
<th>better</th>
<th>equally good</th>
<th>equally bad</th>
<th>total</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>6 (6%)</td>
<td>19 (19%)</td>
<td>28 (28%)</td>
<td>28 (28%)</td>
<td>100</td>
<td>18.13</td>
</tr>
<tr>
<td>NE58 gold</td>
<td>4 (4%)</td>
<td>15 (15%)</td>
<td>28 (28%)</td>
<td>28 (28%)</td>
<td></td>
<td>18.21</td>
</tr>
</tbody>
</table>

Table 4.10: Results of the manual evaluation of baseline run at 1M and NE58 gold with sampling 171:1, both at 1M training steps. The manual evaluation is provided on subsample of newstest2011.

In the 100 randomly selected sentences from newstest2011, which were evaluated, were only 45 sentences with some named entities. See Table 4.11 for the results. Many NE, 73%, were translated the same way in both models, 62% well and 11% badly. The baseline model translated 7 sentences with NE “better” or “substantially better”, while the NE58 gold model only 5. We suppose the baseline is better, but we do not consider this results as significant, because only small number of examples were observed.

### Evaluation Focused to Named Entities

The previous evaluation on newstest did not show any strong results on quality of translation of named entities, because not many relevant examples were observed. Therefore we prepared and evaluated the models on another dataset of

---

3 According to the annotator.
Table 4.11: Results of the manual evaluation of baseline run at 1M and NE58 gold with sampling 171:1, both at 1M training steps. The manual evaluation is provided on subsample of newstest2011.

<table>
<thead>
<tr>
<th></th>
<th>subst. better</th>
<th>Quality of NE:</th>
<th>eq. good</th>
<th>eq. bad</th>
<th>sent. with NE</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>3 (6.67%)</td>
<td>4 (8.89%)</td>
<td>28 (62.28%)</td>
<td>5 (11.11%)</td>
<td>45</td>
</tr>
<tr>
<td>NE58 gold</td>
<td>2 (4.44%)</td>
<td>3 (6.67%)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

100 sentences from the evaluation set of NoSta-D Named Entity Annotation corpus [Benikova et al., 2014] for German. The data come from Wikipedia. We took 20 most common NE tags combinations (compounds of IOB tags, 4 NE classes and two layers, as mentioned in Section 4.1.4) and selected 5 distinct random sentences containing each of them. It is thus ensured that each test sentence contains at least one named entity. We processed the evaluation the same way as before, only the Czech referential translation was not available. Instead of it, the corresponding NE tag sequence was shown to the annotator. See the illustration in Figure 4.15.

source Die Friedberger Landstraße beginnt hier .
ref OxO BuLOCxBuLOCderiv IuLOCxO OxO OxO OxO
b Friedbergerova silnice začíná tady .
Friedbergerova ulice začíná tady .

Figure 4.15: An example of annotation of the sentence “The Friedberger’s Road starts here.” The first sentence is German source. It is followed by compounded NE tags from NoSta-D Named Entity Annotation [Benikova et al., 2014]. Two candidate translations follow. The first is marked as “better”, because it uses a more feasible word road, unlike street in the second translation.

The results of the overall MT quality are summarized in Table 4.12. This evaluation was provided by Marcela Macháčková, a teacher of German and Czech. We assume her professional experience makes her opinions relevant, at least in high scale of 100 examples. This results showed that the baseline is much better than MT+NE58 gold, the score of “better” or “substantially better” examples is 34:20.

Although the first evaluation on newstest showed comparable quality of both settings, we believe the latter is correct. In MT+NE58 translation of the NoSta-D testset (originating in Wikipedia), we find two of 100 sentences ending with several tens of repetitions of the same subword, which are completely nonsense. In the previous newstest translation, we do not find anything similar in substantially higher amount of 3k sentences.

Both these testsets differ in domain. The newstest is consistent set of news articles, while NoSta-D contains sentences from Wikipedia and German online news. The two problematic sentences were about history. We conclude that the baseline model is probably better in translating Wikipedia articles, but the models are comparable in the news domain.

Results of translations of NE are in Table 4.13. Multi-tasking model MT+NE58 gold achieved higher score of “better” or “substantially better” examples,
Table 4.12: Results of the manual evaluation of baseline run at 1M and NE58 gold with sampling 171:1, both at 1M training steps. The manual evaluation is provided on data from NoSta-D Named Entity Annotation corpus [Benikova et al., 2014] for German. Each sentence contains some NE.

25:17. We conclude that the MT+NE58 model sacrifices better translation of named entities for worse overall translation, at least in the Wikipedia domain.

Table 4.13: Results of the manual evaluation of baseline run at 1M and NE58 gold with sampling 171:1, both at 1M training steps. The manual evaluation is provided on data from NoSta-D Named Entity Annotation corpus [Benikova et al., 2014] for German. Each sentence contains some NE.

4.3.3 Baseline vs NE5 Auto Enriching Source

We processed the same manual evaluation as in previous Section 4.3.2 to compare the baseline and MT with enriching source by NE5 auto.

The results of overall MT quality comparison are in Table 4.14. The ratio of “better” or “substantially better” sentences provided by NE5 enriching source and baseline is 22:21. The first is slightly better, but the difference is very small, we do not consider it significant. We conclude the two variants achieve very similar translation quality.

Table 4.14: Results of the manual evaluation of baseline run at 1M and NE5 auto enriching source, both at 1M training steps. The manual evaluation is provided on subsample of newstest2011.

4.3.4 Discussion

In our evaluation, we did not focus on rigorous evaluation of common errors because that would be very demanding and time consuming. Instead of it we
Table 4.15: Results of the manual evaluation of baseline run at 1M and NE58 gold with sampling 17:1, both at 1M training steps. The manual evaluation is provided on subsample of newstest2011.

summarize our subjective observations and opinions about the common flaws, which could be eventually tested by proper evaluation in further work.

State of the Baseline Translation

In our opinion, the baseline translation is already very good and hard to beat. The translation is very adequate, parts of the sentences are rarely omitted. It is also intelligible, the native speaker is able to understand the intended meaning despite some minor flaws in grammar, vocabulary or style. We assume these are not errors, which could not be fixed by commonly used finetuning technique, backtranslation on big in-domain monolingual dataset [Sennrich et al., 2015, Poncelas et al., 2018, Gülçehre et al., 2015], model ensembling [Garmash and Monz, 2016], and with more training time, more training data and bigger model.

Despite of the high baseline, we still see a small space for improvement. Sometimes the model tends to mistranslate an agent and patient. It happens mostly in tricky examples, e.g. when the agent is a proper name, it does not bear a determiner, so the case is not expressed explicitly. See such example on Figure 4.16. We also assume a potential improvement of this issue can not be observed in the difference of BLEU score in standard newstest.

<table>
<thead>
<tr>
<th>model</th>
<th>subst. better</th>
<th>subst. better</th>
<th>eq. good</th>
<th>eq. bad</th>
<th>sent. with NE</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>4 (6.67%)</td>
<td>2 (8.89%)</td>
<td>23 (62.28%)</td>
<td>7 (11.11%)</td>
<td>42</td>
</tr>
<tr>
<td>NE58 gold</td>
<td>3 (4.44%)</td>
<td>3 (6.67%)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.16: Example of a sentence with mistranslated agent and patient.

Common Errors in Named Entities

We observed following kinds of errors in named entities:

- A clause of the source sentence, which accidentally contains also a named entity, is fully omitted in translation.

- Incorrect lemmatization and incorrect NE recognition. In translation from German, we observed some errors in translation of NE in genitive case with case ending “s”. It was misinterpreted as a part of the proper name. Named
entities can be also incorrectly recognized. We illustrate both on German example *Tumba Ottos II.*, which means *Tomb of Otto II.*. It was incorrectly translated to Czech as *Tumba Ottos II.*, as if it was a full personal name.

- Correct lemmatization, but incorrect grammatical category. This is a case of e.g. *v Gambii* versus *v Gambiu*. It means *in Gambia* in Czech. The first is a correct variant in feminine gender, the other is incorrect neuter.

- Incorrect lemmatization and wrong NE category. Sometimes, the MT model misinterprets a place for person and an adjectival suffix for part of lemma. E.g. *Bratislaver Vorstadt* (lit.: *the suburb of Bratislava*), was translated as *Bratislaverovo předměstí* (lit.: *Bratislaver’s suburb*, as if *Bratislaver* was a person’s name).

- Incorrectly translated rare NE, which should be translated. An example is *hessisch*, an adjective form of *Hessen*, one of federal states of Germany. It was translated as *heský*, which is incorrect, but understandable form. The other model translated it as *ruský*, which means *Russian*. The correct form is *hesenský*.

- Incorrectly translated rare NE, which should be partially copied from the source. E.g. *Johannes Nepomuk von Tschiderer* was translated as *Jan *Udpeuk*, or as *Jan Nepomuk z *Čeideru. Correct translation is either the original form, or *Jan Nepomuk z Tschidereru*. 
5. Multi-Tasking with Multiple Decoders

In this chapter, we describe our progress and experiments with implementing support for multiple decoders in OpenNMT-py [Klein et al., 2017]. We selected this framework, because it provides Transformer, its codebase is not so complex as in Tensor2Tensor [Vaswani et al., 2018], it is better documented and easier to extend. Another option would be to use Neural Monkey [Hecl and Libovický, 2017], but we had more experience with OpenNMT so far.

Firstly, in Section 5.1, we find the hyperparameter settings for the new OpenNMT-py baseline, and describe the most critical differences from T2T. In Section 5.2, we review our implementation. In Section 5.3, we describe our experiments with high-resource dataset and big model and their results.

5.1 Transformer Baseline in OpenNMT-py vs Tensor2Tensor

T2T has more configurable parameters than OpenNMT-py and the implementation differs in many details. Therefore, it is impossible to use identical settings and obtain identical results. In this section, we at least find the closest possible setting and state the new OpenNMT-py baseline. Then we explain the differences in performance and suggest improvements.

5.1.1 Closest Possible Setting

Both OpenNMT-py and T2T provide the Transformer model, but they differ in bucketing training examples into batches, in the order of training examples in each epoch, in optimizer options etc.

To find the most similar setting, we started with the one provided in the documentation,\(^1\) which sets the same model size and optimizer parameters as T2T. Then we applied the same batch size 1500 and 60k warmup steps as in T2T, and further set all other hyperparameter and values from T2T, which had configurable parameters in OpenNMT-py.

OpenNMT-py does not provide any implicit method for subword segmentation. To reduce the risk of obtaining different results due to different datasets and segmentation, we preprocessed the data for OpenNMT-py with T2T’s SubwordTextEncoder. We used the identical shared vocabulary of size 100k as for our T2T experiments, justified in in Section 3.3. We transformed the data in preprocessing phase from raw text into indices of vocabulary entries, represented with decimal numbers. We used this approach because OpenNMT-py considers each space-separated word as a single token, but in T2T, some “tokens” can contain spaces. In postprocessing we transform indices back to the text and further evaluate the BLEU scores on full words.

\(^1\)http://opennmt.net/OpenNMT-py/FAQ.html#how-do-i-use-the-transformer-model
In T2T, only the training examples with maximum source and target sequence length equal to the batch size are used. The batch size is usually very high so only a very small number of examples will be filtered out. We set the same maximal sequence length in OpenNMT-py, and for further comparison we report also a run with maximal sequence length 50.

### 5.1.2 Explanation of Different Results

Figure 5.1 shows the validation learning BLEU curve over the training epochs. In T2T, we inferred the approximated number of epochs from average number of non-padding tokens, from batch size and from training steps, as suggested by Popel and Bojar [2018]. OpenNMT-py logs epochs by default. Table 5.1 summarizes the maximal validation BLEU scores obtained.

<table>
<thead>
<tr>
<th>framework</th>
<th>max seq. len.</th>
<th>shards</th>
<th>step</th>
<th>epoch</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tensor2Tensor</td>
<td>1500</td>
<td>-</td>
<td>737620</td>
<td>10.66</td>
<td>18.27</td>
</tr>
<tr>
<td>OpenNMT-py</td>
<td>1500</td>
<td>4</td>
<td>959891</td>
<td>11.24</td>
<td>17.74</td>
</tr>
<tr>
<td>OpenNMT-py</td>
<td>50</td>
<td>5k</td>
<td>1001127</td>
<td>11.72</td>
<td>15.70</td>
</tr>
</tbody>
</table>

Table 5.1: Maximal validation BLEU score achieved by baseline in T2T and OpenNMT-py.

![Validation MT BLEU over training epochs for OpenNMT-py and T2T baseline with closest comparable settings.](image)

We see big spikes corresponding to epochs on the BLEU curve on OpenNMT-py baseline. In OpenNMT-py preprocessing, the training examples are processed in shards of given maximal size. Each time, only one shard is loaded into the working memory (RAM), the rest of training data remain stored at the data storage. The user can set the maximal shard size. Small shard size allocates a small capacity of RAM to prevent out-of-memory errors. On the other hand, we suppose that in the current implementation, the training examples are shuffled and reordered only within the shard, not across the whole data. This reasons in suboptimal bucketing and low-quality shuffling, if the shards are small, which we suppose decreases the overall model performance from BLEU 17.74 to 15.70, as the two OpenNMT-py runs in Figure 5.1 and Table 5.1. However, the actual
behavior of the implementation needs more investigation to confirm or reject this assumption. The loss in performance can be explained also by reducing the training examples by sequence length.

Suggestions for Further Implementation

We suggest the sharding and shuffling should be reimplemented, so it will be assured that the shard sizes do not affect the order and quality of batches. Also, shuffling after each epoch seems not implemented in OpenNMT-py.

5.2 Implementation of Multiple Decoders

We used a very simple, but effective approach to implement support for multiple decoders. Each decoder solves one task, each has its own specific training and validation data. The data are preprocessed, sharded, shuffled and bucketed into batches separately. The vocabulary is shared, created from the union of training datasets from each task.

The the data from distinct tasks are further used separately, they are not mixed into a single training batch. In training, a batch from the main task is generated to perform the optimizer update with the main decoder. Afterwards, the tasks, training data and decoders are swapped to process another batch from secondary task, and so on. The tasks are swapped after each batch. This reduces the risk of adjustment of the encoder to only one of the tasks. The “Trainer” module is responsible for task scheduling and switching decoders. The task identification token appended to training data is thus not necessary.

Our implementation is publicly available on GitHub.\textsuperscript{2} During the implementation, we had to resolve the following issues:

- We had to refactor the codebase to enable preprocessing of two datasets with a shared vocabulary at once, multi-decoder model creation, training, saving to checkpoints, and translation.

- We extended the checkpoint loading and saving, so both multi-decoder and basic models are supported and handled properly.

- The default framework enables checkpoint saving only after each epoch. We extended support for saving after given number of seconds (e.g. each 1 hour or 20 minutes). The checkpoint is stored together with number of “training steps”, which is number of parameter updates after processing one batch, the same way as in T2T.

- We ensured our extensions work on both CPU (for debugging) and GPU (for real training). In GPU, the full model with all decoders must be loaded to memory at creation time.

- We tested our double-decoder implementation on GPU with reduced model size on a small dataset. The “main” task was character-based translation of English proper names to Russian, or, in other words, transliteration from

\textsuperscript{2}https://github.com/Gldkslfmsd/OpenNMT-py
The “secondary” task was uppercasing. The model learned both tasks very well, the BLEU score on 10 validation examples reached almost 100% on both tasks. We achieved the same learning curves even when we swapped the “main” and “secondary” task.

- Memory issues and multi-decoder baseline. The model with two decoders is approximately 50% bigger than with only one decoder. Even with our standard German-to-Czech dataset, we were finally able to fit the double-decoder model with the same internal capacity as the single-decoder baseline (except the second decoder), into 11GB GPU memory. During that, we were getting some out-of-memory errors. We experimented with some hyperparameter values to reduce them, and finally we ended up with the baseline setting and shard size 0.5MB. With these setting, we were able to train the double-decoder model with linguistic tasks up to almost 1.5M steps without interruptions caused by errors.

Despite of the extensive implementation work we did, we still do not consider the double-decoder model as sufficiently robust for training with big datasets. Under certain conditions, which we were not able to explain due to time constraints, some errors inside of the library for computation on GPU still appear. This should be explored and fixed in future work.

We also suggest the framework should be refactored to use the latest version of third-party libraries (pytorch\(^4\) for computation of deep learning models on GPU and CPU, torchtext\(^5\), etc.), which, we assume, could help to fix some of the unexplained errors during training.

Once our implementation will be robust and reliably tested, we plan to create a pull request to merge it into the OpenNMT-py master branch. We assume it will be appreciated by the OpenNMT community.

5.3 Experiments with Multi-Decoder Model

In this section, we experiment with our multi-decoder model for two tasks. The encoder is shared for all tasks, one decoder is dedicated to the MT task and the other one to the secondary linguistic task for processing source. We expect this setting makes encoder more robust because it is forced to encode the source into a form suitable for both tasks.

For simplicity, both the decoders are instances of the identical model, originally designed for MT, with the same capacity (6 layers), although it could be better to use the decoder of a smaller capacity for simpler tasks. We leave this for future work.

5.3.1 Linguistic Secondary Tasks

We use the model for enriching the German source with three linguistic secondary tasks from from Universal Dependencies (Nivre et al. [2017], recall Section 1.3.2):

\(^3\)It is a demonstration dataset from OpenNMT Hackaton: https://github.com/OpenNMT/Hackathon/tree/master/nmt-wizard/data.

\(^4\)https://pytorch.org/

\(^5\)https://github.com/pytorch/text
POS tagging with the universal tagset (UPOS), tagging with dependency labels (Dtags), and with unlabeled dependency parsing sequentialized in a way suggested by Kiperwasser and Ballesteros [2018] (DheadsOffsets). On the position of each source word there is an offset of the absolute position of the word its head in the dependency tree. It is represented as a decimal number, positive, if the head is on the right, negative, if it is on the left. Unlike Kiperwasser and Ballesteros [2018], we denote the root with 0. See illustration in Figure 5.2. The source and target for UPOS and Dtags is similar as in Section 4.1.4.

![Diagram](image)

Figure 5.2: Illustration of the encoding of an unlabeled parsing tree into a sequence of distances. The first row contains the sentence (source) and its parse tree, and the second row contains the matching distances sequence (target). Figure and caption reprinted from Kiperwasser and Ballesteros [2018].

### 5.3.2 Results

Figure 5.3 shows the validation MT BLEU learning rate of the multi-tasking runs and baseline. Figure 5.4 shows the accuracy on the secondary task. Table 5.2 summarizes the quality of the MT and secondary linguistic tasks. For the secondary task we use the same “line accuracy” as in Section 4.2.2. From Figure 5.3 we see that the models are not stable, the validation BLEU curve fluctuates a lot. We want to reduce the risk of reporting single scalar at given number of steps, which can be accidentally inside of a ravine, although neighboring checkpoints have higher scores. We therefore report the average and standard deviation of all checkpoints between 950k and 1.05M training steps. The quality of MT+DheadsOffsets on both MT and parsing is 3 BLEU points worse.

From Table 5.1 it is evident that the multi-tasking with UPOS and Dtags achieved nearly the same MT quality as the baseline. The difference is insignificantly small. The quality of MT+DheadsOffsets is 2.5 BLEU points worse.

The models converged to a constant accuracy of the secondary task very soon, at around 400k training steps.

<table>
<thead>
<tr>
<th>run</th>
<th>MT BLEU</th>
<th>sec. task acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>14.87±0.50</td>
<td>-</td>
</tr>
<tr>
<td>MT+UPOS</td>
<td>14.72±0.55</td>
<td>96.36±0.08</td>
</tr>
<tr>
<td>MT+Dtags</td>
<td>14.83±0.47</td>
<td>89.94±0.10</td>
</tr>
<tr>
<td>MT+DheadsOffsets</td>
<td>12.45±0.36</td>
<td>82.66±0.37</td>
</tr>
</tbody>
</table>

Table 5.2: Validation quality metrics of MT and the secondary task. The reported scores are average±stddev of checkpoints between 950k and 1.05M training steps.
Figure 5.3: Validation MT BLEU over training steps for OpenNMT-py baseline and double-decoder models with secondary linguistic tasks.

Figure 5.4: Validation accuracy of secondary tasks over training steps for double-decoder models.

5.3.3 Discussion

Recall that the models use a very small shard size to avoid out-of-memory errors. The same small shard size is also used in the baseline run, so we assume we can compare it with our multi-tasking models. The shard size can also partially explains the 3.5 BLEU points worse performance than T2T in Section 5.1.

We assume that the POS tagging and dependency parsing do not improve the quality of translation, because they do not provide relevant additional knowledge, which could not be learned from the big amount of data unsupervised. The encoder may work the same way as in the baseline single-task model. The convergence on the secondary task was achieved very soon, so the feedback from the secondary decoder was almost zero and thus not propagated into the encoder. For POS tagging and dependency parsing, we suggest another model implementation, in which only the lower encoder layers are shared. Another suggestion is to experiment rather with named entities or with incorporating semantic knowledge, which requires deeper language understanding and bigger model.

The MT+DheadsOffsets showed bad performance. We assume it is because
the decoder has to learn word boundaries between subwords first, and then pars-
ing. We suggest either experiment with parsing at the subword-level, or imple-
ment, a specific non-sequential decoder for parsing, or predict the head wordform
instead the positional offset, as in Chapter 4 (task Dheads).
Conclusion

In this thesis, we experimented with enriching NMT through multi-task learning on MT and linguistic secondary task. The previous related works were limited to attentional sequence-to-sequence NMT model and relatively small datasets. We focus on comparably bigger datasets and on the newest state-of-the-art Transformer model, whose adaptation for multi-tasking has not been implemented or explored yet, but is considered as very promising.

We promote the NMT model knowledge of source side morphology (POS tagging), syntax (dependency parsing) and named entities. Primarily, we experiment with a simple multi-tasking architecture where all the model components are shared for all tasks, the multiple tasks are determined by simple data manipulation technique, and the training alternates between the tasks.

We evaluate our experiments with automatic metrics, and compare the linguistically enriched multi-tasking models with the baseline single-task MT and multi-tasking models with dummy linguistically unrelated secondary tasks. We show, that for the standard big data settings (89M and 78M source and target side tokens, 8.8M sentence pairs, German-to-Czech), automatically annotated datasets and the same amount of primary and secondary task training data, the enriched models outperformed the models with dummy secondary tasks. This is an evidence, that the multi-tasking with enriched source can be efficiently utilized by the NMT model. However, we also show the that in the multi-tasking model with alternating setup, the multi-tasking cost decreases the overall translation quality. On the other hand, in small data setting with 500k sentence pairs the model for MT and dependency labeling significantly outperformed the single-tasking baseline by 2 BLEU points.

Two of our experiments incorporating named entities achieved the same BLEU score as the baseline. Our manual evaluation on 100 sentences showed that they achieve either the same, or insignificantly worse overall translation quality, and the quality of translation of named entities is comparable, and in one case much better than in the baseline model.

Finally, we aimed at implementing multi-decoder Transformer model in OpenNMT-py framework. We overcame many technical issues and were finally able to run experiments with joint models for MT and POS tagging, dependency labeling and unlabeled dependency parsing for big data scenario. The first two models achieved a comparable MT quality as the baseline.

The secondary goal of our work was to set the new state-of-the-art in German-to-Czech machine translation, which has received a limited research interest in recent years. We compounded previously published parallel corpora, cleaned and preprocessed them into a new dataset for German-to-Czech MT.

This work is a continuation of our previous study “Morphological and Language-Agnostic Word Segmentation for NMT” [Macháček et al., 2018], which was accepted to the 21st International Conference on Text, Speech and Dialogue (TSD 2018), and will be published there in September 2018. In this study [Macháček et al., 2018] we explored the methods for subword segmentation of morphologically rich languages, namely German and Czech. We compared two linguistically uninformed methods, byte-pair-encoding and SubwordTextEncoder,
and two morphology-based methods, DeriNet, a novel method based on derivational morphology, and Morfessor. The methods were validated on the same German-to-Czech dataset, as we use in this thesis.
Bibliography


Ondˇ rej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina, editors. *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Associa-


Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of


List of Figures

1.1 BLEU scores for English-Spanish systems trained on 0.4 million to 385.7 million words of parallel data. Quality for NMT starts much lower, outperforms SMT at about 15 million words, and even beats a SMT system with a big 2 billion word in-domain language model under high-resource conditions. Figure and caption reprinted from Koehn and Knowles [2017]. ........................................... 6

1.2 Illustration of the basic RNN Encoder–Decoder NMT model. Figure reprinted from https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-2/. ............ 8

1.3 Illustration of attentional encoder–decoder NMT model. Figure reprinted from https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/. ........... 9

1.4 The Transformer model architecture. Figure and caption reprinted from Vaswani et al. [2017]. ................................................................. 11

1.5 The Vauquois Triangle. Figure reprinted from https://sites.google.com/site/b490nlp/machine-translation ......................... 13

1.6 A screenshot from Google Translate. Left: The Czech source sentence “Mum grinds meat.” in all acceptable word orderings. Right: English translations by Google Translate. We see that only the most common ordering is translated correctly. ................. 13

1.7 German sentence I have a cat. annotated with Universal Dependencies. Visualization of syntactic dependency tree on top, table with all annotations at the bottom. Example provided by http://lindat.mff.cuni.cz/services/udpipe/. ............... 14

1.8 Relations between layers in PDT. The rendered Czech sentence Byl by ˇ sel dolesa. (lit.: He-was would went to forest.) contains past conditional of the verb jít (to go) and a typo *dolesa instead of do lesa. Figure and example reprinted from https://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/en/html/ch02.html. ............ 15

2.1 Overview of different architectures for attentional sequence-to-sequence NMT multi-task learnings. Figure and caption reprinted from Niehues and Cho [2017]. ................................. 18

2.2 Illustration of different scheduling strategies determining the probability of the next training example to be picked from each of the multiple tasks we learn. Each sub-plot in the figure matches a different scheduling strategy (with $\alpha$ set to 0.5). The sub-plot describes the probability ($p$, y-axis) of the task we wish to improve ($q$) using Scheduled Multi-Task Learning as a function of the number of epochs trained ($t$, x-axis) so far. The remaining probability is uniformly distributed among the rest of the tasks. Figure and caption reprinted from Kiperwasser and Ballesteros [2018]. ....... 20

2.3 Integrating target syntax in the NMT decoder. Left: interleaving, right: multitasking. Figure and caption reprinted from Nadejde et al. [2017]. ................................. 22
2.4 Original dependency tree for the sentence “Leonidas begged in the arena.”, and its feature representation after BPE segmentation. Figure and caption reprinted from Sennrich and Haddow [2016].

3.1 Examples of low quality parallel sentences in Europarl.

4.1 Example of multitasking variants: alternating, interleaving and enriching input.

4.2 Example of variants of named entity annotation of German source sentence “Besides, he published two songbooks, Songs From The Hills Of Tennessee and Arthur Smith’s Original Song Folio No.1.” (NE columns) and the original NoSta-D 2 layer annotation, which we transformed into NE58 gold.

4.3 Example of linguistic additional tasks.

4.4 Example of synthetic dummy tasks.

4.5 Validation BLEU on MT task over training steps on best runs for each secondary task. Baseline is single-task MT.

4.6 Validation BLEU on MT task over MT epoch on best runs for each additional task. One epoch has 89M/78M tokens of source and target, respectively, or 8.7M parallel sentences. Baseline is single-task MT.

4.7 Validation BLEU on MT task over training steps for small and big dataset. Baseline is single-task MT.

4.8 Validation BLEU over training steps for comparison of vocabulary design techniques on different kinds of linguistic and synthetic additional tasks.

4.9 Validation line accuracy of linguistic (bold lines) and dummy secondary tasks (thin lines) over training steps.

4.10 Validation BLEU of MT+linguistic (bold lines) and dummy secondary tasks (thin lines) over training steps.

4.11 Validation MT BLEU over training steps for adaptation of MT+Dheads models to MT.

4.12 Validation MT BLEU over training steps for experiments with sampling of MT and Dheads training data for double-tasking. All runs have identical vocabulary.

4.13 Validation MT BLEU over training steps for experiments with sampling of MT and Dheads training data for double-tasking. All runs have an identical vocabulary.

4.14 An example of annotation of the sentence Uli Gsell also got some of them. In total around 400 kilograms. The first sentence is German source, the second is Czech referential translation. Two candidate translations follow, the first one is marked as “substantially better” and in fact is provided by the model at 250k steps.

4.15 An example of annotation of the sentence “The Friedberger’s Road starts here.” The first sentence is German source. It is followed by compounded NE tags from NoSta-D Named Entity Annotation [Benikova et al., 2014]. Two candidate translations follow. The first is marked as “better”, because it uses a more feasible word road, unlike street in the second translation.
4.16 Example of a sentence with mistranslated agent and patient. 61

5.1 Validation MT BLEU over training epochs for OpenNMT-py and T2T baseline with closest comparable settings. 64

5.2 Illustration of the encoding of an unlabeled parsing tree into a sequence of distances. The first row contains the sentence (source) and its parse tree, and the second row contains the matching distances sequence (target). Figure and caption reprinted from Kiperwasser and Ballesteros [2018]. 67

5.3 Validation MT BLEU over training steps for OpenNMT-py baseline and double-decoder models with secondary linguistic tasks. 68

5.4 Validation accuracy of secondary tasks over training steps for double-decoder models. 68
List of Tables

3.1 Training corpora sizes before and after cleanup ........................................ 27
3.2 Final training data statistics. ................................................................. 27

4.1 Size of training MT vocabulary before segmentation and size of target side vocabularies of additional tasks. .................................................. 42
4.2 Summary of all experimental runs using T2T at 600k training steps or the nearest available checkpoint. BLEU scores and average numbers of subwords are reported on validation data. Abbreviations of experimental runs are explained in Section 4.1. The symbols d/y/n/- denote “default”, “yes”, “no”, and “not applicable”, respectively. .................................................. 45
4.3 Contrastive summary of small and big training datasets for German-to-Czech translation. ................................................................. 46
4.4 Contrastive summary of experimental runs with the small and big dataset. BLEU scores and average numbers of subwords are reported on validation data. .................................................. 47
4.5 Summary of all experimental runs using T2T at 600k training steps or at nearest available checkpoint. BLEU scores and average numbers of subwords are reported on validation data. Abbreviations of experimental runs are explained in Section 4.1. .................................................. 50
4.6 Validation quality metrics of secondary tasks. Top: Scores for classification, precision, recall, and F1 with macro averaging, accuracy and baseline (accuracy of predicting most common class). Bottom: mean squared error and mean absolute error for regression tasks (counting or enumerating correct count of words or subwords). .................................................. 52
4.7 Validation MT BLEU and parsing quality for task sampling experiments. ................................................................. 54
4.8 Validation MT BLEU and NE tagging quality for experiments with named entities. Accuracy (acc.) and baseline accuracy (base.) is reported. Note that for NE gold and automatic data, different datasets are used. Sampling parameter is the ratio of MT and secondary task training data. .................................................. 56
4.9 Results of manual evaluation of the baseline run at 1M and 250k training steps on randomly selected 78 sentences from newstest2011. BLEU score is reported on the whole newstest2011. .................................................. 58
4.10 Results of the manual evaluation of baseline run at 1M and NE58 gold with sampling 171:1, both at 1M training steps. The manual evaluation is provided on subsample of newstest2011. .................................................. 58
4.11 Results of the manual evaluation of baseline run at 1M and NE58 gold with sampling 171:1, both at 1M training steps. The manual evaluation is provided on subsample of newstest2011. .................................................. 59
4.12 Results of the manual evaluation of baseline run at 1M and NE58 gold with sampling 171:1, both at 1M training steps. The manual evaluation is provided on data from NoSta-D Named Entity Annotation corpus [Benikova et al., 2014] for German. Each sentence contains some NE. .................................................. 60
4.13 Results of the manual evaluation of baseline run at 1M and NE58 gold with sampling 171:1, both at 1M training steps. The manual evaluation is provided on data from NoSta-D Named Entity Annotation corpus [Benikova et al., 2014] for German. Each sentence contains some NE. .................................................. 60
4.14 Results of the manual evaluation of baseline run at 1M and NE5 auto enriching source, both at 1M training steps. The manual evaluation is provided on subsample of newstest2011. ............. 60
4.15 Results of the manual evaluation of baseline run at 1M and NE58 gold with sampling 171:1, both at 1M training steps. The manual evaluation is provided on subsample of newstest2011. ............. 61

5.1 Maximal validation BLEU score achieved by baseline in T2T and OpenNMT-py. .................................................. 64
5.2 Validation quality metrics of MT and the secondary task. The reported scores are average±stddev of checkpoints between 950k and 1.05M training steps. .................................................. 67
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>artificial intelligence</td>
</tr>
<tr>
<td>BLEU</td>
<td>bilingual evaluation understudy</td>
</tr>
<tr>
<td>BPE</td>
<td>byte pair encoding</td>
</tr>
<tr>
<td>CCG</td>
<td>combinatorial categorial grammar</td>
</tr>
<tr>
<td>CPU</td>
<td>central processing unit</td>
</tr>
<tr>
<td>cs</td>
<td>Czech</td>
</tr>
<tr>
<td>de</td>
<td>German (deutsch)</td>
</tr>
<tr>
<td>DT</td>
<td>dependency tagging</td>
</tr>
<tr>
<td>en</td>
<td>English</td>
</tr>
<tr>
<td>GB</td>
<td>gigabyte</td>
</tr>
<tr>
<td>GPU</td>
<td>graphic processing unit</td>
</tr>
<tr>
<td>IWSLT</td>
<td>International Workshop on Spoken Language Translation</td>
</tr>
<tr>
<td>k</td>
<td>thousand (after a number, e.g. 500k)</td>
</tr>
<tr>
<td>LM</td>
<td>language model</td>
</tr>
<tr>
<td>M</td>
<td>million (after a number, e.g. 4.5M)</td>
</tr>
<tr>
<td>MB</td>
<td>megabyte</td>
</tr>
<tr>
<td>MT</td>
<td>machine translation</td>
</tr>
<tr>
<td>NE</td>
<td>named entity</td>
</tr>
<tr>
<td>NER</td>
<td>named entity recognition</td>
</tr>
<tr>
<td>NLP</td>
<td>natural language processing</td>
</tr>
<tr>
<td>NMT</td>
<td>neural machine translation</td>
</tr>
<tr>
<td>NN</td>
<td>neural network</td>
</tr>
<tr>
<td>PBMT</td>
<td>phrase-based machine translation</td>
</tr>
<tr>
<td>PDT</td>
<td>Prague Dependency Treebank</td>
</tr>
<tr>
<td>POS</td>
<td>part of speech</td>
</tr>
<tr>
<td>RNN</td>
<td>recurrent neural network</td>
</tr>
<tr>
<td>SMT</td>
<td>statistical machine translation</td>
</tr>
<tr>
<td>STE</td>
<td>SubwordTextEncoder</td>
</tr>
<tr>
<td>T2T</td>
<td>Tensor2Tensor</td>
</tr>
<tr>
<td>TI</td>
<td>task identification</td>
</tr>
<tr>
<td>UD</td>
<td>Universal Dependencies</td>
</tr>
<tr>
<td>UPOS</td>
<td>universal part of speech</td>
</tr>
<tr>
<td>WIT³</td>
<td>Web Inventory of Transcribed and Translated Talks</td>
</tr>
<tr>
<td>WMT</td>
<td>Workshop on Machine Translation</td>
</tr>
</tbody>
</table>
A. Attachments

A.1 Content of the Electronical Attachment

In this electronical attachment, we include data, which the reader may use for further analysis and replication of our experiments. Due to the space constraints, we can not include e.g. any binary model parameter files, all translated validation and test files for experiments, etc. We include only the following:

Thesis Text


• thesis-simplex.pdf – this thesis in PDF format, single-page typesetting, 85 pages. This version is more suitable for reading on electronical devices.

Datasets

• de2cs_dataset – the German-to-Czech parallel corpus used in this work

Experiment Results

• experiments/ – all translated files for the baseline and the two enriched models, which achived comparable BLEU score as the baseline

• dev-600k-bigtable/ – translated validation files by all reported experiments in T2T at 600k training steps

• onmt-table/ – analogical for the OpenNMT-py experiments

• man-eval-decs/ – sources and results of the manual evaluation

Collections of Scripts and Implementation

• t2t_scripts/ – scripts used for the T2T experiments

• MT-ComparEval-master/ – an extension of MT-ComparEval [Klejch et al., 2015] and scripts, which we used to generate the tables and plots included in this thesis

• OpenNMT-py-master/ – fork of the OpenNMT-py [Klein et al., 2017] repository with our implementation of the multi-decoder model