Karlsruhe Institute of Technology

Target Factors for Neural Machine Translation

Diploma Thesis of

Martin Wagner

at the KIT Department of Informatics
Institute for Anthropomatics and Robotics (IAR)

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Prof. Dr. Tamim Asfour
Advisor:	Dr. Jan Niehues

01. April 2017 – 30. September 2017

Karlsruhe Institute of Technology
KIT Department of Informatics
Postfach 6980
76128 Karlsruhe

I declare truthfully that I have developed and written this thesis by myself, that I have completely declared all sources and means from other pieces of work, which I quoted with or without modification, and that I comply to the current charter of the KIT regarding good scientific practice.

Karlsruhe, 9/30/2017

. . . . . .

(Martin Wagner)

Abstract

Neural Machine Translation has recently shown promising results, replacing Statistical Machine Translation as state-of-the-art approach to machine translation.

The following work examines the usefulness of integrating *Target Factors* into an attentional encoder-decoder Neural Machine Translation system in order to improve translation quality. Target Factors are all kinds of, mostly linguistically motivated, information that can be gathered about the target language and that can be expressed on wordlevel. Factors utilized in this work include part of speech tags enriched with morphosyntactic information (RFTags), lemmas, Dependency Labels and IOB tags together with sbuword units based on BPE segmentation. In order to derive this additional information, external tools are used to annotate the target sentences of the bilingual training corpus.

The approaches for integrating this additional information into an existing Neural Machine Translation system presented in this work include an *N-Best List Re-Ranking* study, a *Serialization* approach and the joint prediction of the target factors in every decoding step (*Joint Decoding*). Moreover, it is examined whether the application of such factors to the target language in combination with the factorization of the source language, already known from literature, can be beneficial to the translation process. Additionally we deployed the serialization approach for the source language as well.

The approaches are evaluated on a low-resource translation task. The results obtained for the N-Best List Re-Ranking study are promising and the Serialization approach leads to slight improvements up to +0.7 Bleu when used with a modified lemma vocabulary. Regarding the Joint Decoding approach, no improvements could be discerned, which is consistent with other results released recently in literature. Concerning the combination of factorizing source and target language, significant improvements of +0.9 Bleu can be observed. This result was achieved when the proposed serialization approach is applied for both the source and the target language.

Zusammenfassung

Neuronale maschinelle Übersetzung ist ein derzeit vielversprechender Forschungsgegenstand auf dem Gebiet der maschinellen Übersetzung. Die vorliegende Arbeit befasst sich mit der Untersuchung, ob die Integration von sogenannte *Target Factors* in ein neuronales maschinelles Übersetzungssystem die Übersetzungsqualität zu verbessern vermag. Target Factors, zu deutsch *zielsprachliche Faktoren*, bezeichnen prinzipiell jegliche Art der, die Zielsprache betreffenden zusätzlichen Informationen, für die gilt, dass sie sich auf Wortebene darstellen lassen. Um sie in das Training des Systems einfließen lassen zu können, werden in dieser Arbeit die Trainingsdaten auf Wortebene mit den eingesetzten zusätzlichen Faktoren annotiert. Um dies zu ermöglichen, wird hierbei aufgrund von Verfügbarkeit und Qualität hauptsächlich auf linguistisch motivierte Faktoren und die dazu vorhandenen Tools zurückgegriffen. In dieser Arbeit werden um reichhaltige morphosyntaktische Information erweiterte Part Of Speech Tags (RFTags), Lemmata, Dependency Labels und Information über die Subwortstruktur (IOB Tags) als zusätzliche Faktoren neben den Subworteinheiten, auf denen das Basissystem operiert, eingesetzt.

Die vorgestellten Ansätze zur Integration dieser zusätzlichen Information in ein bestehendes neuronales maschinelles Übersetzungssystem beinhalten eine *N-Best Listen Re-Ranking* Studie, einen Serialisierungsansatz (*Serialization*) und ein Verfahren um die Faktoren in jedem Dekodierschritt gemeinsam vorherzusagen (*Joint Decoding*). Darüberhinaus wird untersucht, ob ihr Einsatz in der Zielsprache zusammen mit der bereits in der Literatur bekannten Faktorisierung der Quellsprache vorteilhaft ist. Dabei wird auch der Serialisierungsanatz als alternative Möglichkeit der Quellsprachenfaktorisierung vorgeschlagen.

Die Ansätze wurden auf einem “low-resource task” evaluiert, wobei sich zeigte, dass die Ergebnisse der N-Best Listen Re-Ranking Studie vielversprechend waren, und dass der Serialisierungsansatz leichte Verbesserungen von bis zu +0,7 Bleu mit einem modifizierten Lemmata Vokabular erreichen konnte. Für den Joint Decoding Ansatz konnten keine Verbesserungen festgestellt werden, was im Einklang mit jüngst erschienener Literatur steht. Für die Kombination der Faktorisierung von sowohl Ziel- als auch Quellsprache konnten klare Verbesserungen von etwa +0.9 Bleu nachgewiesen werden. Dieses Ergebnis konnte erzielt werden, indem für die Faktorisierung von sowohl Quell-, als auch Zielsprache der vorgeschlagene Serialisierungsansatz verwendet wird.

Contents

Abstract iii

Zusammenfassung v

1. Introduction 1
- 1.1. Motivation . . . 2
- 1.2. Objective . . . 5
- 1.3. Outline . . . 5

2. Background 7
- 2.1. Neural Machine Translation (NMT) . . . 7
  - 2.1.1. Attentional RNN Encoder-Decoder Architecture . . . 8
  - 2.1.2. Word Representation . . . 10
  - 2.1.3. Recurrent Neural Network (RNN) . . . 12
  - 2.1.4. RNN Encoder – Decoder . . . 13
  - 2.1.5. Gated Recurrent Units (GRUs) . . . 14
  - 2.1.6. Attention Mechanism . . . 15
  - 2.1.7. Large Vocabularies: Byte Pair Encoding (BPE) . . . 17
  - 2.1.8. Training . . . 18
  - 2.1.9. Translation using Beam Search . . . 19
- 2.2. Factors and External Tools . . . 20
  - 2.2.1. Parts-of-Speech and Morphosyntactic Information . . . 20
  - 2.2.2. Lemmas . . . 25
  - 2.2.3. Dependency Labels . . . 25
  - 2.2.4. IOB Tags . . . 26
- 2.3. BLEU Evaluation Metric . . . 26

3. Related Work 31
- 3.1. Factors in Machine Translation . . . 31
  - 3.1.1. Statistical Machine Translation (SMT) . . . 31
  - 3.1.2. Neural Machine Translation (NMT) . . . 33

4. Target Factors for Neural Machine Translation 37
- 4.1. Assumptions . . . 37
- 4.2. Levels of Component Sharing . . . 38
- 4.3. Proposed Approach . . . 39
  - 4.3.1. N-Best List Re-Ranking . . . 39
  - 4.3.2. Serialization of Target Factors . . . 40

4.3.3. Joint Decoding of Target Factors . . . 43
4.3.4. Combination With Source Factorization . . . 45

5. Evaluation **47**
5.1. Experimental Setup . . . 47
5.1.1. Neural Machine Translation Toolkit . . . 47
5.1.2. Data and Hyper-parameters . . . 47
5.1.3. Evaluation Method and Baseline . . . 51
5.2. Experiments and Results . . . 53
5.2.1. N-Best List Re-ranking . . . 53
5.2.2. Serialization of Target Factors . . . 55
5.2.3. Joint Decoding of Target Factors . . . 60
5.2.4. Combination With Source Factorization . . . 62
5.3. Discussion . . . 65

6. Conclusion **69**
6.1. Summary . . . 69
6.2. Future Work . . . 70

References **71**

A. Appendix **79**
A.1. N-Best List Re-ranking . . . 79
A.2. Serialization of Target Factors . . . 81
A.3. Joint Decoding of Target Factors . . . 87
A.4. Combination With Source Factorization . . . 93
A.4.1. Source Factorization . . . 93
A.4.2. Combination of Source and Target Factor Approaches . . . 96

1. Introduction

Anytime a
linguist
leaves the group
the recognition rate goes up.
— Frederick Jelinek[1]

We live in a globalized world where means of translation are an essential cornerstone of human interaction. Growing international cooperation and the multiplicity of worldwide spoken languages increase the need to bridge language barriers.

The interconnection through the World Wide Web enables one common pool of information, the exchange of user-generated content on Web 2.0 platforms and collaborative work around the globe by multinational corporations, thereby promoting cultural exchange and mutual learning. This interaction requires the understanding of different languages. The human ability to learn foreign languages is naturally limited and a large scale deployment of human translators is not feasible in these scenarios, therefore shifting the focus to automated solutions. The research field of machine translation (MT) focuses on the computer-based automation of the translation process from one natural human language (*source language*) into another (*target language*) with no human translator involved.

Neural machine translation (NMT) is a research field recently introduced for machine translation. It has shown very promising results and emerges to replace state-of-the-art statistical machine translation (SMT) systems (e.g., Bentivogli et al., 2016; Junczys-Dowmunt et al., 2016). Within a few years after its first introduction NMT is already deployed in commercial applications (Crego et al., 2016; Wu et al., 2016).

The idea of NMT is to use an artificial neural network to manage the entire translation process, which will be explained in detail in Section 2.1. NMT comprises several advantages, the most notable are the following: NMT offers a holistic solution. A single, jointly trained model replaces the variety of loosely coupled weak feature functions which were deployed in SMT. The network learns continuous word representations that map semantically or syntactically related words to nearby points in a word embedding space. This allows for similar words to share knowlegde as opposed to the former SMT approaches which treated words independently whenever their surface form[2] differed. Moreover, these representations are gained by the system itself, thus relieving the developer of the intricate task of rich feature design while surpassing the quality of hand designed features. Lastly, the use of neural networks with recurrent connections leads to deep structures that allow the consideration of a larger context in order to model long range dependencies among words.

[1] cf. Jurafsky and Martin (2009)

[2] the form in which the word or sentence is written in the text corpus

NMT is getting closer to human translation quality, but the latter is not yet met (Wu et al., 2016). Therefore, further improvements require extensive research.

1.1. Motivation

Neural machine translation is based on a general sequence-to-sequence learning approach that treats sentences as sequences of generic tokens. Thereby, it does not explicitly make use of linguistic properties of the languages involved. Therefore, the question arises whether providing additional information, e.g., in form of linguistic annotations, can help to improve translation performance. Recently Sennrich and Haddow (2016) successfully enriched the *source* language sentences with linguistic features. A natural extension to this approach is to take the *target* language into consideration.

Therefore, this thesis investigates the introduction of *target factors* to neural machine translation. A factor refers to "each type of additional word-level information" (Koehn, 2009). Thus, we define *target* factors as any type of additional word-level information related to the *target* language exclusively. Word level factors comprise linguistic information, for example part of speech tags, lemmas or morphosyntactic labels like case, gender or number. But they also include rather abstract information like statistically derived word classes or tags that annotate sub-word structure (see Section 2.2). In principle, factors could be any kind of automatically derivable information that is representable at word level. External tools are used to acquire the annotations for the training data corpus.

An important difference between deploying factors for the target language as opposed to the source language includes the fact that source factors can not only be inferred during training, but at translation time as well. Whereas the target factors can not be inferred during translation, because the target sentence is not given during translation. Utilizing external tools to obtain the additional factors on a *partial* hypothesis for the target sentence is discouraged because we assume it to be both error-prone and slow. Therefore, the NMT system can only profit from the annotations at training time and has to generate them during the inference phase itself.

Figure 1.1 depicts the basic setup. Note that the ultimate goal of translation still is to generate a valid target sentence based on words. The additional factors are only used internally and are discarded when the final hypothesis is output to the user. However, they might be useful during the translation process allowing the decoder to make better choices concerning the generation of target words.

The various motivations to include additional factors in general presented in the literature include the following aspects (e.g., Hoang et al., 2016; Nadejde et al., 2017; Sennrich and Haddow, 2016). An important reason to use additional information is data sparseness. Lemmatization can help especially for morphologically rich languages, in which certain morphological variants are rare and therefore robustly estimated representations cannot be easily found. But when the base form (i.e., the lemma) of a word is additionally used, knowledge might be shared among all its inflectional variants, allowing for better generalization. Further improvements that might be expected concern the word order and word agreement of the target language. Thereby additional knowledge about a broader

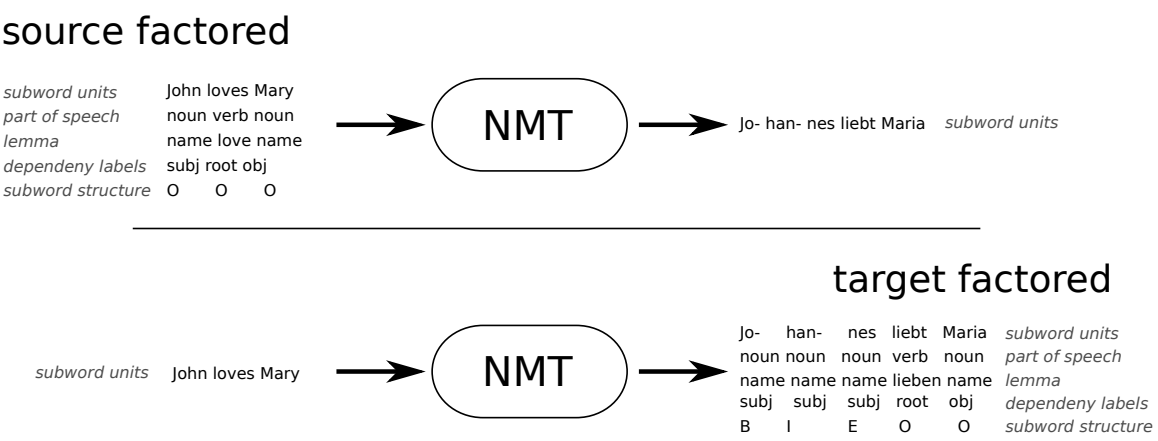


Figure 1.1.: Overview of source and target factorization. This is a fictive example. The number and the choice of additional factors is arbitrary. Further description of the factors used in this work is found in Section 2.2. Note that a combination of both approaches is possible as well, respective experiments are performed in Chapter 5.

categorization like parts of speech and the morphological attributes of consecutive words might be helpful to estimate the correct choices.

Additional information might also help to disambiguate homographs, i.e., words that portray different meanings although sharing a common surface form. This is especially helpful for the source language, because as homographs differ in meaning they often result in different translations, which makes disambiguation necessary. The motivation to enclose additional information on target side for disambiguation purposes is slightly different. It stems from the concern to enable different meanings of a homograph to be expressed with different internal representations. As long as only the surface form of a word is used, it inevitably results in the same internal representation, but when including additional factors their representations can be combined to form overall representations that allow for disambiguation. Table 1.1 shows an example where two different meanings of the English word *can* might be disambiguated by their part of speech (verb compared to noun). Also when using sub-word units[3] depending on the design choices made for sub-word modeling an end unit might be ambiguous to a whole word (e.g., San Francis*can* in Table 1.1). In this case IOB Tags[4] that encode the sub-word structure can support disambiguation. This allows for three different internal representations of the word *can* instead of only a single one.

The use of external tools to derive the additional factors includes the following benefits. Dependent on their design they might be trained on large amounts of monolingual data that significantly surpass the amount of bilingual data which is used to train the NMT model. Moreover handcrafted rules can be exploited, leveraging the language expertise of linguist experts. At least the tools' design might be influenced by task-specific knowledge, e.g. the part of speech of an unknown word might be successfully guessed by its suffix

[3] we use BPE sub-word units described in Section 2.1.7
[4] see Section 2.2.4

(Schmid, 1994a). Lastly, for an under-resourced language pair it might be easier to build simple taggers and parsers based on handwritten rules than acquiring larger amounts of bilingual data.

Table 1.1.: Example for different interpretations of the word "can"

verb	Alice **can** sing .
noun	I enjoy a **can** of beer .
sub-word (end-)unit	My father was a San Franc- is- **can** .

The concept of word factorization was originally introduced to the field of machine translation by Koehn and Hoang (2007) when building factored translation models for statistical machine translation. Sennrich and Haddow (2016) pose the question of how much of their findings can be transferred to NMT. The latter has stronger implicit learning capabilities than SMT. In SMT any linguistically motivated concept to be learned must be explicitly represented as sequences of words in surface form, which in consequence leads to data sparsity problems because not all relevant word combinations that cover the linguistic concept can be represented in a training data corpus. Therefore, falling back to a more general representation than words in surface form, e.g., parts of speech or lemmas, is assumed to be beneficial. In NMT however, those concepts can, to some extend, already be implicitly learned from the lexical data (Shi et al., 2016). This could make explicit provision redundant and therefore supplant the necessity of additional target factors.

Sennrich and Haddow (2016) postulate that these effects will be even stronger the more training data is available. In this regard, as a rather qualitative consideration to be mentioned, we observe a comparable human behavior. While toddlers learn their mother tongue without the explicit knowledge about any linguistic concepts (which an average child is probably confronted with in school for the first time), they nevertheless develop a form of *mind's ear* that can implicitly discourage wrong grammar usage and therefore make the explicit knowledge about linguistic concepts superfluous. On the other side, a non-native speaker who learns a foreign language is confronted with loads of linguistic information, mostly in form of grammar and morphology rules. This procedure of course might be related to different language learning abilities at different ages. Still, it is believed that also the low resource setting a foreign language learner usually is confronted with is a driving force for this approach, allowing to quickly explorate the language space without the help of constant parental feedback.

Concerning NMT, at the time of writing, data is assumed to be sparse either due to low resource settings where the amount of available data is simply too small to develop syntactically and semantically rich models. In this case we expect additional information to be helpful. But also because of the lack of deep semantic understanding that NMT still suffers from (e.g., Bentivogli et al., 2016), this pattern matching approach cannot correctly translate unknown idioms, metaphors and other constructs with no comparable observation in the training data as for this purpose, higher levels of meaning need to be extracted. However, to what extent rather abstract and low level linguistic knowledge can help in these situations is underexplored.

1.2. Objective

The objective of this work is to find empirical evidence whether integrating additional knowledge about the target language into an Attentional Encoder-Decoder Neural Machine Translation system leads to translation performance improvements. The following additional knowledge sources are to be investigated. As linguistic information, part of speech tags with rich morphosyntactic annotation, lemmas and syntactic dependency labels, and as structural information IOB tags, are used. All additional factors are derived by existing external tools for the target side of the training data corpus. Different possible solutions to augment the target words with additional factors are to be examined and compared to similar approaches taken in recent literature.

1.3. Outline

The remaining thesis is structured in the following way:

In Chapter 2, the theoretical foundations of this work regarding neural machine translation are given. We thereby focus strongly on the current state-of-the-art attentional RNN encoder-decoder architecture. Moreover the factors used in this work are described. We provide comprehensive overview of the RFTagger, as the morphological enriched part of speech tags it provides originally were our main additional factor. We however chosen to include lemmas, dependency labels and IOB tags for better comparison and therefore we give brief description. As for evaluation purposes the BLEU metric is used we give a summary of the latter.

In Chapter 3, previous approaches using factorization in machine translation are presented.

In Chapter 4, the integration of target factors for neural machine translation is discussed. We state or different approaches and motivate our design decisions.

Chapter 5 contains the evaluation results of the proposed approaches.

Finally, this work is concluded in Chapter 6 with a short summary and outlook.

2. Background

This thesis investigates the integration of additional knowledge implemented in the form of word-level factors into a Neural Machine Translation system. Therefore, a brief overview on Neural Machine Translation is given in this chapter with strong focus on the Attentional Recurrent Neural Network (RNN) Encoder-Decoder architecture.

Subsequently, the features used for word factorization will be presented along with the external tools to derive them. The proposed methodology does not depend on any specific functional principles of those tools. However, it relies on the availability and correct operation of the latter, because the approach taken in this thesis requires respective factor annotations for the bilingual training corpus. As gold annotations by human annotators are usually not available for a bilingual training corpus, we have to resort to automatic tools.

This chapter concludes with a presentation of the Bilingual Evaluation Understudy (BLEU), the most widespread automatic evaluation metric for machine translation.

2.1. Neural Machine Translation (NMT)

Neural Machine Translation (NMT) is a recently proposed approach to machine translation. NMT attempts to model the entire translation process with a single, large artificial neural network (ANN) which is jointly trained in an end-to-end fashion on a bilingual corpus. Instead of integrating neural-network-based components into an existing machine translation approach, NMT provides a stand-alone translation system (Zhang and Zong, 2015).

From a probabilistic perspective, neural machine translation seeks for a parametric model that provides a conditional probability $P(y \mid x)$ for a target sentence y given a source sentence x. This goal is common to statistical machine translation. Brown et al. (1993) define a frequentist interpretation to $P(y \mid x)$ as the probability that a human translator would translate x to y.

The general translation problem is described as finding the most likely target language sentence $\hat{\mathbf{y}}$ given a source language sentence $\mathbf{x} = (x_1, x_2, \ldots, x_{L_x})$ of length L_x. Using a maximum likelihood approach, $\hat{\mathbf{y}}$ can be found by solving Equation 2.1, thereby using the chain rule of conditional probabilities (Goodfellow et al., 2016), which is useful to build up the target sentence incrementally.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\, P(\mathbf{y}|\mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmax}} \prod_{i=1}^{L_y} P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) \tag{2.1}$$

It is important to note that this probability is modeled directly. No assumptions are made, no externally enforced simplifications are introduced by design. This implies that the

whole context is taken into account and no Markov assumption has to be made to cut it off (Kalchbrenner and Blunsom, 2013).

Compared to previous approaches, this is a more generic approach to translation. Instead of designing dedicated feature functions or translation rules, it is modeled as generic sequence-to-sequence learning problem (Cho et al., 2014b; Sutskever et al., 2014).

2.1.1. Attentional RNN Encoder-Decoder Architecture

Using neural networks for the task of machine translation poses some challenges which are addressed by the components of the state-of-the-art Attentional RNN Encoder-Decoder Architecture.

A simplified sketch of this architecture is depicted in Figure 2.1.

Firstly, words are discrete tokens that need to be suitably represented in a neural net, which only operates on real-valued vectors of fixed dimension (Cho et al., 2014a), discussed in Section 2.1.2.

Secondly, the input and output layers of neural networks are of fixed size while natural language sentences are of variable length. This problem is tackled by the Encoder-Decoder Architecture (Cho et al., 2014a; Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014). In the current state-of-the art both, encoder and decoder are based on Recurrent Neural Networks (RNN), described in Section 2.1.3. Alternatively, Convolutional Neural Networks (CNNs) (Kalchbrenner and Blunsom, 2013; Waibel et al., 1989) were successfully introduced to be used as encoder, but are not covered in this work.

Thirdly, different parts of the source sentence are usually translated to different parts of the target sentence. How the neural network is given the ability to learn a proper mapping is shown in Section 2.1.6.

All these challenges are solved in the Attentional RNN Encoder-Decoder Architecture, which is depicted in Figure 2.1.

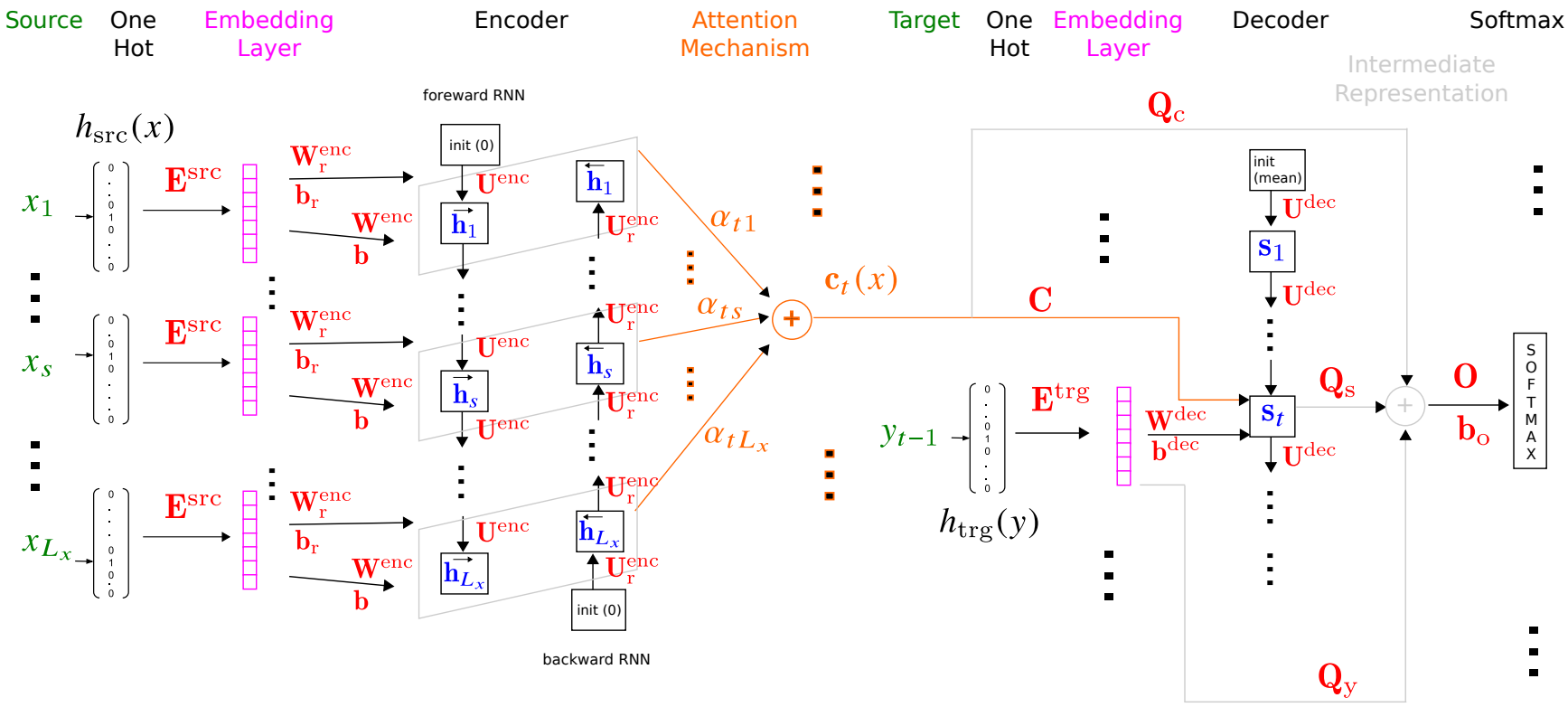


Figure 2.1.: Overview of the Attentional RNN Encoder-Decoder Architecture used in the Neuronal Machine Translation approach.

2.1.2. Word Representation

Let us first define the input representation of words suitable for the neural network to operate on. It is thereby necessary to have input layers for source words at the encoder and target words at the decoder. Words are discrete tokens, that can be unambiguously represented by their index in the respective source or target vocabulary. A neural network on the other hand operates on real valued vectors. To define an internal representation that is well suited for the translation process would be a cumbersome task if done manually. But this is actually not a challenge but a benefit of the NMT approach that it is able to find internal word representations of remarkable quality itself as described in the following. We thereby first need a representation that ensures the words in the vocabulary to be mathematically truly independent in terms of vector distances. This is not the case for using vocabulary indices, e.g., when using alphabetical ordering in the english vocabulary the distance of indices of the words "muscle" and "museum" is small though these words are not closely related semantically. Therefore the one-hot encoding is used to ensure independence.

One-Hot Encoding In One-Hot Encoding, each word in a vocabulary W is represented by a $|W|$-dimensional vector with almost all entries set to zero. Only a single entry identifying the k-th word $w_k \in W$, is set to one. The feature vector length corresponds with the vocabulary size $|W|$. The ordering of words in the vocabulary must be defined once and remain fixed but is otherwise arbitrary. This representation is usually chosen as word input for the neuronal network (Lebret, 2016). We will denote the one-hot vector of a word index $x \in W$ as $h(x)$. Distinct vocabularies are used for the source and target language.

Word Embedding To use One-Hot encoded input is already a sufficient premise for the neural network to learn its own internal representation by the error backpropagation learning procedure. In order to investigate these internal representations we can use an optional word embedding layer. Its introduction has more advantages explained below. From a mathematical viewpoint the embedding layer is similar to other layers used in the architecture with slight modifications. Its activation functions are linear and a bias vector is usually not used. The dimension of the embedding layer is chosen to be significantly smaller than the size of the respective input word vocabulary. To compute the embedding layer the input vector given in One-Hot coding, representing the k-th vocabulary word, is projected linearly by a respective weight matrix $\mathbf{E}$:

$$e(x) = \mathbf{E}h(x), \quad \mathbf{E} \in \mathbb{R}^{d \times |W|} \tag{2.2}$$

As in this special case the input is a One-Hot vector and no bias is used, the multiplication equals the selection of the k-th column in the weight matrix. Mathematically this equals an projection or *embedding* of a higher dimensional vector space into lower dimensional pendant. $\mathbf{E}$ is also called embedding matrix and $e(x)$ embedding vector, the latter span the embedding space. The embedding matrix is gained by the joint training of the network automatically. Because the dimension of the embedding space is chosen to be substantially smaller than the One-Hot space, the network is forced to find a representation of increased

efficiency, as words cannot be represented completely independent anymore. Moreover the latter is enabled by a property of natural language. It is known, that words occurring in similar contexts show relations both semantically and syntactically. This property was already successfully used to cluster words in an unsupervised manner (Brown et al., 1992). The novelty of word embeddings is that for a word multiple relations expressed by different dimensions in the embedding space can be learned and expressed in a real valued space instead of learning affiliations to discrete classes (Mikolov et al., 2013a).

For a different but related architecture and learning task Mikolov et al. (2013a,b,c) could show that when using billions of words in a monolingual corpus remarkable dependencies among words can be learned to be represented in the embedding space. It could be shown that nearby embedding vectors, expressed by vector distances like euclidean distance or similarity measures like cosine similarity are semantically and syntactically related. Also linear relationships among embedding vectors could be used to solve analogical reasoning tasks (Mikolov et al., 2013b). Figure 2.2 shows an example of how the relation “gender” can be expressed in the embedding layer as linear vector offset. One could use the difference between the vectors of “man” and “woman” and add this difference vector to the vector representation of “king” to find out that the feminine form of the latter is “queen”[1].

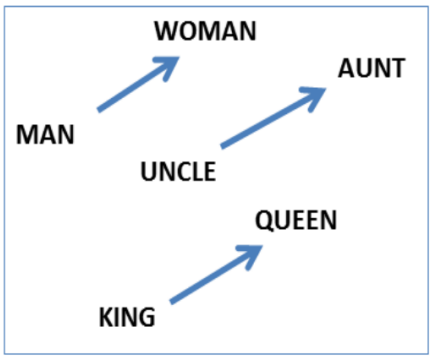


Figure 2.2.: Gender relation expressible as linear vector offset in the embedding space, from Mikolov et al. (2013c)

Even phrases and whole sentences can be expressed with internal vector representations (Cho et al., 2014b).

The main goal of machine translation is however not to solve analogical reasoning tasks, but rather to find high quality translations. Also the embeddings will in that respect be of lower quality as usually less bilingual data is available than monolingual data. However when the system learns how words relate we assume the data sparseness problem to be alleviated as formerly distinct words may share knowledge which in consequence will improve the generation capabilities. Moreover embedding quality might even partly increase by disambiguation of words on the basis of translation choice. Yang et al. (2013) could show that the antonyms "good" and "bad" are highly related in word representations trained with monolingual data, as they appear in similar contexts. But when using bilingually trained embeddings their representations diverge.

[1]by a nearest neighbor search in the vicinity of the resulting position in the embedding space

To explicitly use embedding layers has the advantage that they can be tied in the case of using multiple encoders (e.g., the BiRNN, presented in Section 2.1.6) and that when using multiple factors an embedding can be learned for each with distinct dimension which are then combined appropriately as shown in Section 4.3.3.

2.1.3. Recurrent Neural Network (RNN)

RNN is a rich dynamic model for sequence learning tasks. It has been successfully used as language model. We first state the general formulation and then show how RNNs can be used as encoder and decoder.

As shown in Figure 2.3, a RNN consists of an input layer $\mathbf{w}$, a hidden layer $\mathbf{h}$ and a optional output layer $\mathbf{o}$ (Mikolov et al., 2010). The hidden layer is a self-recurrent, i.e., a copy of its output at the last time step $t-1$ is used as input at the current time step t (Goodfellow et al., 2016).

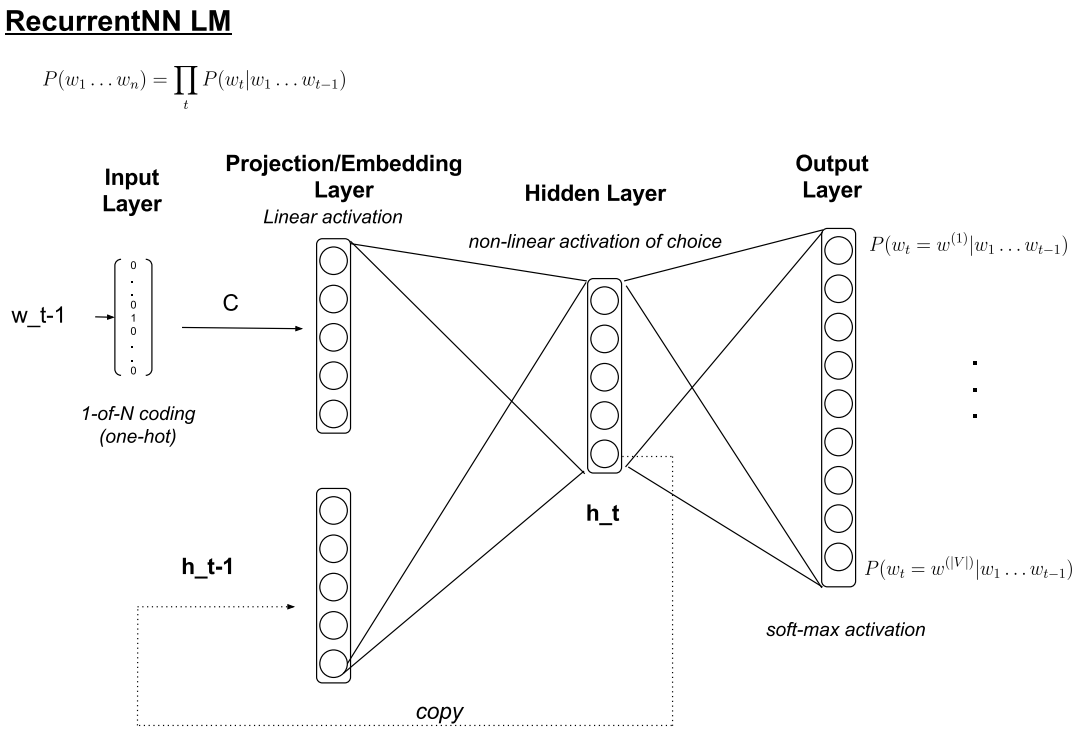


Figure 2.3.: A Recurrent Neural Network which processes a phrase word-by-word.

Equation 2.3 shows how the hidden layer $\mathbf{h}_t$ in time step t can be calculated: First, the t-th word is represented using the One-Hot Encoding $\mathbf{w}_t$. Then, it is projected into the Word Embedding $e(\mathbf{w}_t)$ where $e(\cdot)$ represents the projection function. Together with the output of the hidden layer from the last time step $\mathbf{h}_{t-1}$, an arbitrary activation function $f(\cdot)$ calculates the output of the hidden layer from the current time step. For the first hidden

layer $\mathbf{h}_1$, $\mathbf{h}_0$ is set to zero. As an example, Equation 2.3 uses the hyperbolic tangent as activation function, a linear projections applied on $\mathbf{h}_{t-1}$ and $e(\mathbf{w}_t)$ represented by weight matrices $\mathbf{U}$ and $\mathbf{W}$ and bias $\mathbf{b}$. (Cho et al., 2014a; Mikolov et al., 2010)

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, e(\mathbf{w}_t)) \overset{\text{e.g.}}{=} \tanh(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}e(\mathbf{w}_t) + \mathbf{b}) \tag{2.3}$$

Equation 2.4 presents the computation of the RNN's output layer: The output can be calculated by any function $g(\cdot)$ using the output of the hidden layer $\mathbf{h}_t$, but usually a multinomial probability distribution over all words is desired as output form so that the softmax function is used in most cases. Equation 2.5 shows how each vector component of the softmax function is calculated.

$$\mathbf{o}_t = g(\mathbf{h}_t) \overset{\text{e.g.}}{=} \text{softmax}(\mathbf{O}\mathbf{h}_t + \mathbf{b}_o) \tag{2.4}$$

$$\text{softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.5}$$

The architecture for NMT uses modified RNNs as both encoder and decoder which are described in the next section.

2.1.4. RNN Encoder – Decoder

The basic idea is to first digest a complete source sentence using an Encoder, resulting in an intermediate representation and then generate the target sentence using a Decoder based on the intermediate representation (Cho et al., 2014b).

Encoder The encoder RNN is not connected directly to the output layer. Instead, the hidden encoder states are used to compute the input which to fed into the decoder. This can be done by simply using the last encoder hidden state as it contains a *summary* of the source sentence (Cho et al., 2014b) or by using the attention mechanism described in Section 2.1.6. The encoder is depicted in Figure 2.1.

The encoder formula to calculate the hidden layer's output is the same as presented in Section 2.1.3 (Equation 2.3). Only the output layer is omitted.

Decoder The decoder is also an RNN and it basically represents a Neural Language Model (NLM) for the target language with slight modifications:

Obviously, information about the source language sentence must be included. This can be done once by initializing the decoder's hidden layer with information about the source sentence, as extra input to each decoding step or a combination of both.

Lastly, Cho et al. (2014b) proposes to make the decoder's output layer dependent on the last target word and the information about the source sentence.

The decoder is also depicted in Figure 2.1. The calculations to update the hidden state $\mathbf{s}_t$ after generating a word y_{t-1} is described in Equation 2.6: Using the last decoder state $\mathbf{s}_{t-1}$, a target language embedding projection $e_{\text{trg}}(\cdot)$, last generated word y_{t-1} and the context of the entire source sequence $\mathbf{c} = \mathbf{h}_t$, which is the output of the encoder in the last time step and an activation function $f^{\text{dec}}(\cdot)$, the current decoder state $\mathbf{s}_t$ can be computed. In

the example of Equation 2.6, the hyperbolic tangent is used as activation function and the input vectors are projected linearly using the matrices $\mathbf{U}^{\text{dec}}$, $\mathbf{W}^{\text{dec}}$, $\mathbf{C}$ and bias $\mathbf{b}_{\text{h}}^{\text{dec}}$.

$$\mathbf{s}_t = f^{\text{dec}}(\mathbf{s}_{t-1}, e_{\text{trg}}(y_{t-1}), \mathbf{c}) \overset{\text{e.g.}}{=} \tanh(\mathbf{U}^{\text{dec}}\mathbf{s}_{t-1} + \mathbf{W}^{\text{dec}}e_{\text{trg}}(y_{t-1}) + \mathbf{C}\mathbf{c} + \mathbf{b}_{\text{h}}^{\text{dec}}) \tag{2.6}$$

Using the hidden state for the current target word $\mathbf{s}_t$, an intermediate representation for the current target word $\mathbf{q}_t$ can be computed using Equation 2.7 (Sennrich et al., 2017):

$$\mathbf{q}_t = \phi(\mathbf{s}_t, y_{t-1}, \mathbf{c}) \overset{\text{e.g.}}{=} \tanh(\mathbf{Q}_{\text{s}}\mathbf{s}_t + \mathbf{Q}_{\text{y}}e_{\text{trg}}(y_{t-1}) + \mathbf{Q}_{\text{c}}\mathbf{c} + \mathbf{b}_{\text{q}}) \tag{2.7}$$

Finally, the output vector $\mathbf{o}_t$ representing a probability distribution over all possible target words to be outputted of decoder step t can be computed using Equation 2.8:

$$\mathbf{o}_t = g(\mathbf{q}_t) \overset{\text{e.g.}}{=} \text{softmax}(\mathbf{O}\mathbf{q}_t + \mathbf{b}_o) \tag{2.8}$$

In summary, the RNN Encoder-Decoder can estimate the probability of target sentence $\mathbf{y}$ by converting the whole source sentence $\mathbf{x}$ into a continuous representation $c(x)$, as depicted in Equation 2.9. The main advantage of RNNs is that context of arbitrary length can be modeled.

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{L_y} P(y_t|\mathbf{c}(\mathbf{x}), y_1, ..., y_{t-1}) \tag{2.9}$$

Note that the length of the target sentence does not have to be known in advance. The target sentence will be terminated as soon as the decoder outputs a special end-of-sentence (<eos>) token, making the prediction of the target sentence length a decision of the model. During training it is therefore necessary to finalize each target sentence in the training corpus with the <eos> token in order for the model to learn this behavior. The search procedure to derive the most likely target word sequence given a source sentence is described in Section 2.1.9.

To further improve the training, Gated Recurrent Units (GRUs) can be used as activation functions. GRUs are able to "forget" the context, which is useful as not all context is relevant for translating parts of the sentence.

2.1.5. Gated Recurrent Units (GRUs)

From a mathematical view, RNNs with conventional units suffer from the vanishing gradient problems during training. As the recurrences are unrolled for training to form a feed forward like topology, the depth of the net is high. Therefore, the gradients underflow with increasing depth of history, effectively limiting the history on which the nodes can be conditioned. Long-Short-Term-Memory units (LSTM) are said to cope with the vanishing gradient problem (Hochreiter and Schmidhuber, 1997).

Gated recurrent units, proposed by Cho (2015), are a simplification of LSTM. As shown in Figure 2.4, the update gate $\mathbf{z}$ selects whether the hidden state is to be updated with a new hidden state $\tilde{\mathbf{h}}$. The reset gate $\mathbf{r}$ decides whether the previous hidden state is ignored. Update and reset gates should not be considered as binary vectors but as real-valued

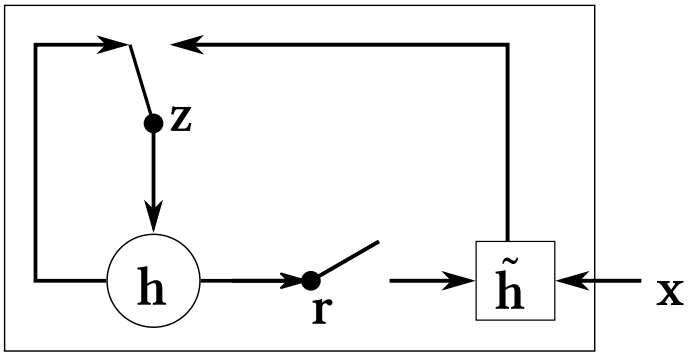


Figure 2.4.: A Gated Recurrent Unit, from Cho et al. (2014b).

coefficient vectors: $z \in [0,1]^{|h|}, r \in [0,1]^{|h|}$. The following equations 2.10 – 2.13 describe how the hidden state $\mathbf{h}$ is computed using $\mathbf{r}$, $\mathbf{z}$ and $\tilde{\mathbf{h}}$ (Cho et al., 2014b):

$$\text{reset gates:} \quad \mathbf{r} = \sigma\left(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}\right) \tag{2.10}$$

$$\text{update gates:} \quad \mathbf{z} = \sigma\left(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}\right) \tag{2.11}$$

$$\tilde{\mathbf{h}}_t = \phi\left(\mathbf{W}\mathbf{x}_t + \mathbf{U}\left(\mathbf{r} \odot \mathbf{h}_{t-1}\right)\right) \tag{2.12}$$

$$\mathbf{h}_t = \mathbf{z} \odot \mathbf{h}_{t-1} + \left(\vec{1} - \mathbf{z}\right) \odot \tilde{\mathbf{h}}_t \tag{2.13}$$

GRUs are said to be able to condition on things at arbitrary points in the source sentence. From a more intuitive point of view, a system that is able to adaptively forget and remember has the advantage that it does not have to model the entire history with all its hidden units but may rather focus only on the relevant parts. This is especially useful when deploying the attentional mechanism described in the next section.

2.1.6. Attention Mechanism

In the previous sections, we described that it is feasible to encode the entire source sentence into a vector of fixed length, e.g., the last hidden state of the encoder and to use this information as input to a decoder which generates the target sentence (Sutskever et al., 2014). Thereby, the vector containing all the sentence information might be used to initialize the decoder or as additional input in every decoding step. Cho et al. (2014a) and Pouget-Abadie et al. (2014) report a decreasing translation performance with increasing sentence length. Sutskever et al. (2014) suggest a heuristic by reversing the word order of source sentences. They claim that when encoding the source sentence's first words just before the first words of the target sentence are decoded, the generation of the latter will benefit from the introduced short term dependencies. This might be at least the case for languages sharing a common word order. However, it is assumed to be cumbersome to compress all the expressiveness a natural language sentence may exhibit including syntactic phenomena like word ordering into a single vector of limited length. Also, since the beginnings of statistical machine translation, it could be shown that decent translation performances can be reached when utilizing word-to-word (Brown et al., 1993) or phrase-to-phrase correspondences (Koehn, 2009).

Therefore, a more sophisticated approach to NMT is the attention mechanism introduced by Bahdanau et al. (2014). Instead of compressing the complete source sentence into a single vector of limited expressiveness, the generation of each target word shall rather focus only on necessary parts of the source sentence relevant for translation of that target word. Instead of falling back on discrete representations like words or phrases, the so far presented architecture based on continuous representations can still be used with only slight modifications.

Until now, the encoder is altered to also be able to encode information about the remaining source sentence at each encoding step. The task is not only to encode information about previous words: At each encoding step, the encoder must be able to encode information about the entire sentence, i.e., also about words still lying ahead. That is why in Figure 2.1, the encoder is depicted as bi-directional RNN (BiRNN), comprised of a forward RNN and a backward RNN. The latter is conceptually the same as the former, but simply reads the source sentence from its end to its beginning. The hidden states in each time step from the forward encoder RNN and the backward encoder RNN are concatenated and are conveniently called annotations. Then, the weighted sum of all annotations from each time step is computed as input for each decoder step.

The weights are learned by jointly training of the entire network. The design of weight computation must not introduce a dependency on the length of the source sentence. As the weights do not depend on each other, a normalization step is performed to make them sum to one.

Using recurrent units with memory behavior, like LSTMs or GRUs, in combination with the decoder states attending only to relevant encoder states allows the encoder to encode only information about the source word whenever its context is not necessary for translation. Whereas this design might perform simple word-to-word translations, it is also able to consider longer contexts if necessary.

In their model architecture Bahdanau et al. (2014) define

$$p\left(y_i \mid y_1, \ldots, y_{i-1}, x\right) = g\left(y_{i-1}, s_i, c_i\right) \ , \tag{2.14}$$

where s_i is the RNN hidden state for time i, computed by

$$s_i = f\left(s_{i-1}, y_{i-1}, c_i\right) \ . \tag{2.15}$$

For each target word y the probability is conditioned on a distinct context vector c_i, which depends on a sequence of annotations $(h_1 \ldots h_{T_x})$. The input sentence is mapped to this annotations by the encoder. Then, the context vector is calculated as weighted sum of the annotations:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \ . \tag{2.16}$$

The weight α_{ij} of each annotation h_j is computed by:

$$\alpha_{ij} = \frac{exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} exp\left(e_{ik}\right)} \ , \tag{2.17}$$

where $e_{ij} = a\left(s_{i-1}, h_j\right)$ is an alignment model. It scores the matching between inputs at position j and outputs at position i, based on the RNN hidden state s_{i-1} and the j-th annotation h_j of the input sentence (Bahdanau et al., 2014).

As a BiRNN consists of a forward and backward RNN,

- the forward RNN $\overrightarrow{f}$ is ordered from x_1 to x_{T_x}, reads the input sequence and computes a sequence of forward hidden states $\left(\overrightarrow{h_1} \ldots \overrightarrow{h_{T_x}}\right)$,
- the backward RNN $\overleftarrow{f}$ is ordered from x_{T_x} to x_1 and computes a sequence of backward hidden states $\left(\overleftarrow{h_1} \ldots \overleftarrow{h_{T_x}}\right)$.

By concatenating both hidden states, the annotation for each word is achieved:

$$h_j = \left[\overrightarrow{h_j^T}, \overleftarrow{h_j^T}\right]^T \quad . \tag{2.18}$$

2.1.7. Large Vocabularies: Byte Pair Encoding (BPE)

A drawback of NMT currently is that full-blown vocabularies of natural languages do not fit into current memory of modern GPUs (Garcia-Martinez et al., 2016a). Therefore, vocabulary size must be reduced. Splitting words into sub-word units is a way to do that. Encoding rare and unknown words as sequences of sub-word units also helps to deal with data sparseness and rare, unknown vocabulary (Sennrich et al., 2015b). It has also the advantage that for morphological rich languages, morphological variants of words can be generated, that do not occur in the training data. E.g., this is the case with the virtually unlimited number of composite words in German.

Sub-word Translation using Byte Pair Encoding The basic idea is that words can be translated using smaller units therefore allowing open-vocabulary translation with a fixed vocabulary and was proposed by Sennrich et al. (2015b). Byte Pair Encoding (BPE) originally is a simple textual data compression method: Byte pairs are merged into symbols and symbols are recursively merged into higher level symbols such that no symbol occur more than twice or there is no unused byte anymore.

Transferred to the word segmentation problem a similar approach is taken. Instead of merging frequent pairs of bytes, the authors merged characters or character sequences. The algorithm can be implemented as straightforward bottom up clustering and is depicted in Figure 2.5. In general it operates on clusters that are represented each by a character string. It starts at character level with each character found in a training corpus to represent an own cluster. In every iteration of the clustering procedure those two clusters are combined where the concatenation of their character strings is most frequent on the training data. A new cluster is inserted represented by this concatenation witch is dubbed as a *merge*. Thereby its parent clusters are not deleted. A special character is inserted at the end of each word in order to ensure that merges are not performed across word boundaries. If the algorithm converged, every single word in the training data would be represented

by a cluster. Therefore usually the algorithm is terminated when the number of clusters roughly equals the desired vocabulary size. For segmentation purposes the learned merge rules are then applied to the data, which is therefore split down to single characters which are subsequently merged to reveal the sub-word units.

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

Figure 2.5.: BPE segmentation algorithm, from Sennrich et al. (2015b).

Applying BPE segmentation to source and target language was called *joint BPE*. Thereto, a common set of merge rules is learned on the joint set of training data of both languages. This will ensure that words common to both languages like proper names will be modeled the same way.

2.1.8. Training

To train the presented model, a stochastic gradient descent based algorithm in conjunction with back-propagation is used to estimate the model parameters (cf. Goodfellow et al., 2016). For back-propagation, the RNN is unfolded through the time steps. To calculate the gradient, the output of the decoder, starting from the input, needs to be differentiable to get a differentiable objective function for the optimization problem associated with the learning process (Cho et al., 2014b).

Cho et al. (2014b) formulate the training problem as maximizing the conditional log-likelihood in Equation 2.19, which is a maximum likelihood approach (Wu et al., 2016). In Equation 2.19, θ represent the model parameters, i.e., the weight matrices and biases and each (y_n, x_n) is an input/output sentence pair from the training set (Cho et al., 2014b).

Maximizing Equation 2.19 is equal to minimizing the cross-entropy, which is proportional to the negative log-likelihood of the otherwise identical equation.

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(\mathbf{y}_n|\mathbf{x}_n) \tag{2.19}$$

Training the network is accelerated by a Graphical Processing Unit (GPU) using mini batches that fit into the memory of a graphics card.

Drawbacks of end-to-end Training One drawback of end-to-end training is that it is bound to the bilingual data corpus, which is limited. For most languages a lot more monolingual data exists. The latter has been used to train language model components vital to traditional SMT systems. But the inclusion of monolingual data or any other kind of data, as for instance the linguistically annotated data the external tools used in this work was build upon, is not straightforward.

2.1.9. Translation using Beam Search

There are different possibilities to use the trained encoder-decoder in order to translate an unseen input sentence. The source sentence is first encoded by the encoder conceptually the same way as done in the forward propagation during training.

To use the decoder to generate a translation hypothesis iteratively word by word, several search strategies are possible:

1. use the most probable target word generated in each decoding step ("greedy decoding" (Stahlberg, 2016)),
2. sample a word from the probability distribution over the target vocabulary at each decoding step ("stochastic sampling"),
3. take all combinations of all possible target words into account leading to different paths in a search graph. Then apply the Beam Search pruning technique to make this approach feasible.

In each of the above presented attempts a hypothesis (path) is finalized when the *<eos>* symbol[2] is chosen by the search strategy based on the decoder's output.

The third case mentioned above is the common practice. The purpose of pruning is to boost time efficiency taking into account a degradation of decoding performance. Beam search uses a pruning rule to select only promising nodes in a path at each level of the search graph by a heuristic (Zhou and Hansen, 2005). The extend and complexity of the search are determined by varying the beam width. In NMT normally a fixed beam size is used. It was empirically shown that small beam sizes are sufficient. Often a fixed beam size of 12 or less is utilized (Garcia-Martinez et al., 2016b; Sutskever et al., 2014).

The beam search procedure in NMT is straightforward and can be implemented as follows. A fixed number of k hypotheses is tracked where k is the beam size. For each tracked

[2] end-of-sequence symbol

hypothesis the next decoding step is performed resulting in a probability distribution over all vocabulary words to be the next word on that hypothesis. Each hypothesis is then expanded by all the words in the vocabulary leading to $k|W|$ new hypotheses in total where $|W|$ is the size of the target vocabulary. Overall scores are calculated and the k best new hypotheses are kept discarding all others. The procedure is iterated for subsequent decoding steps. Thereby the total number of decoding steps is not known in advance. The decoder itself decides when to finalize a hypothesis by predicting the <eos> symbol. For each finalized hypothesis the beam width is reduced. The search procedure comes to an end when all remaining hypotheses are finalized or a maximum number of decoding steps is reached.

2.2. Factors and External Tools

This section contains a brief presentation of automatically derivable additional knowledge which is used in this work as additional factors to augment the target language sentences.

This work only considers additional knowledge which can be represented on word level. Also, all additional factors should be generically interchangeable and not requiring any special factor-dependent treatment such as a factor-dependent topology of the NMT system. These requirements are not met by syntactic parse trees as defined by *phrase structure grammars* or *dependency grammars* which represent the syntactic structure of a sentence on a higher level than word level.

The automatic derivation of additional factors should not be restricted to a narrow domain which currently discourages semantic labeling, especially for the target language German.

2.2.1. Parts-of-Speech and Morphosyntactic Information

Part of speech (POS) is a categorization of words based on their syntactic role in a sentence (Radford, 2006). POS indicates a class of words that occur in similar positions or fulfill similar functions. The main POS in English are noun, pronoun, adjective, determiner, adverb, verb, preposition, conjunction, and interjection (Collins English Dictionary, 2017). Parts of speech are not to be confused with the syntactic functions itself, e.g., subject, predicate, object.

POS tagging refers to the assignment of the correct part of speech label (subsequently referred to as *POS tag*) to each token in a sentence. Tokens comprise words, numbers, punctuation marks and special character sequences. A *tagset* defines which POS tags to consider and may vary largely among different POS taggers, mostly due to varyingly vigorous sub-categorization of part of speech classes (e.g., adjectives might be further specified to be *attributive adjectives* or *adjective with predicative or adverbial usage*).

Regarding neural machine translation, POS tags could help to disambiguate homographs, at least in such cases where differences in semantic content lead to different parts of speech. An example of such ambiguities in German is given in Table 2.1. Furthermore, POS tagging can be seen as a form of word classification which might help to share information about words with the same POS tag. Both mentioned advantages are already achieved by NMT

systems to great extend by utilizing information about context and translation choice to disambiguate and (soft-)classify words.

Table 2.1.: Example for homographs in German. Part of Speech tags help to disambiguate the different meanings.

noun	pronoun	adjective	verb	cardinal number
		die **faulen** Kinder (the lazy children)	Äpfel **faulen** (apples rot)	
Sein (existence)	**sein** Name (his name)		müde **sein** (be tired)	
Regen (Rain)		Im **regen** Verkehr (in busy traffic)	Schnecken **regen** sich nicht (Snails don't move)	
Lauten *pl.* (lutes)		Lärm durch **lauten** Gesang (noise due to loud singing)	die Gewinner **lauten** (the winners are (called))	
			Wir **sieben** Sand (we screen sand)	**sieben** Zwerge (seven dwarfs)

However, because bilingual training data is sparse, it is assumed that words with few occurrences cannot be represented robustly. POS taggers on the other hand can be trained on monolingual data which might exceed the amount of bilingual data. Note that in the approach taken in this work, a word must nevertheless appear in the bilingual corpus along with its POS tag(s), generated by an external POS tagger, in order for the NMT system to learn this mapping likewise, i.e., knowledge about words which are not present in the bilingual corpus can not be transferred into the NMT system.

POS tagging itself could be done by classification of words in isolation. But the complexity of tagging derives from the fact that the POS tags to be assigned to a word often are ambiguous. Besides, a tagger designed for open domain usage must be able to deal with *unknown words*, i.e., words that were not foreseen in the taggers vocabulary but are encountered during application. The exploration of context may be helpful to alleviate both problems (Koehn, 2009). Therefore tagging is to be seen as a sequence labeling approach rather than a classification of words in isolation (Neunerdt, 2016).

Statistical approaches to POS tagging follow the basic formula rendered in Equation 2.20: The most probable tag sequence $\hat{t}_1^N$ shall be found by maximizing the a-posteriori probability $p\left(t_1^N | w_1^N\right)$ given the word sequence w_1^N. Either this probability is modelled directly or Bayes' law may be used to achieve the transformations displayed in Equation 2.21. The denominator may be omitted as $p(w_1^N)$ remains constant for all possible tag sequences. The remaining numerator is equivalent to the joint probability of tag and word sequences.

$$\hat{t}_1^N = \underset{t_1^N}{\operatorname{argmax}}\, p(t_1^N | w_1^N) \tag{2.20}$$

$$= \underset{t_1^N}{\operatorname{argmax}}\, \frac{p(w_1^N | t_1^N) p(t_1^N)}{p(w_1^N)} = \underset{t_1^N}{\operatorname{argmax}}\, p(w_1^N | t_1^N) p(t_1^N) = \underset{t_1^N}{\operatorname{argmax}}\, p(t_1^N, w_1^N) \tag{2.21}$$

One instance of statistical approaches is to use a *Hidden Markov Model* (HMM) to approximate this probability as described by Equations 2.22 to 2.25. After rearrangement of the joint probability in Equation 2.23 (Charniak et al., 1993) the following two simplification assumptions are made:

Firstly, the k-th order Markov assumption $p(t_i|t_1^{i-1}, w_1^{i-1}) \approx p(t_i|t_{i-k}^{i-1})$ lets each state depend only on its k predecessor states, but each state is independent from any other states as well as from any emissions (Du Preez and Weber, 1998).

Secondly, the output independence assumption $p(w_i|w_1^{i-1}, t_1^i) \approx p(w_i|t_i)$ states that the current observation w_i depends on the current state t_i only and should be independent of all previous observations and states (Huang et al., 2001).

$$p(t_1^N, w_1^N) = p(t_1, ..., t_N, w_1, ..., w_N) \tag{2.22}$$

$$= p(t_1)p(w_1|t_1)p(t_2|t_1, w_1)p(w_2|t_1, t_2, w_1) \ldots p(t_N|t_1^{N-1}, w_1^{N-1})p(w_N|t_1^N, w_1^{N-1}) \tag{2.23}$$

$$= \prod_{i=1}^{N} p(t_i|t_1^{i-1}, w_1^{i-1})p(w_i|t_1^i, w_1^{i-1}) \tag{2.24}$$

$$\approx \prod_{i=1}^{N} p(t_i|t_{i-k}^{i-1})p(w_i|t_i) \tag{2.25}$$

Schmid and Laws (2008) and Charniak et al. (1993) see in this result that the probability of the tags depend on a lexical and a contextual factor, as depicted in Equation 2.26.

$$p\left(t_1^N, w_1^N\right) = \prod_{i=1}^{N} \underbrace{p\left(t_i \mid t_{i-k}^{i-1}\right)}_{\text{context prob.}} \underbrace{p\left(w_i \mid t_i\right)}_{\text{lexical prob.}} \tag{2.26}$$

The contextual probability $p\left(t_i \mid t_{i-k}^{i-1}\right)$ could be simply estimated by using n-grams, e.g., Brants (2000) proposes trigrams in Equation 2.27.

$$\text{Trigrams:} \quad \hat{P}\left(t_3 \mid t_1, t_2\right) = \frac{\mathrm{f}\left(t_1, t_2, t_3\right)}{\mathrm{f}\left(t_1, t_2\right)} \tag{2.27}$$

The lexical probability $p\left(w_i \mid t_i\right)$ is modeled by the emission probabilities and can be estimated by counting the number of co-occurrences of tags and words, normalized by the number of tag occurrences on the training data. As an example, Brants (2000) uses the Equation 2.28 to calculate the probability.

$$\text{Lexical:} \quad \hat{P}\left(w_3 \mid t_3\right) = \frac{\mathrm{f}\left(w_3, t_3\right)}{\mathrm{f}\left(t_3\right)} \tag{2.28}$$

RFTagger In this work, the RFTagger by Schmid and Laws (2008) is used to annotate the training corpus with linguistic knowledge. It is a probabilistic HMM tagger and its performance could be shown to be superior or at least comparable to other tagging approaches (Clematide, 2013; Maier et al., 2014; Sajjad and Schmid, 2009). The German version was trained on parts of the TIGER corpus (Brants et al., 2002) which consists

of German newspaper sentences with high quality linguistic annotations undertaken or corrected by human annotators. The tagset of the TIGER corpus is defined upon POS base categories derived from the *Stuttgart Tübingen TagSet* (STTS) of about 50 tags (Schiller et al., 1995), which Schmid and Laws (2008) redefined to 23 POS categories. These base tags are further enriched by the morphosyntactic attributes *number*, *gender*, *case*, *person*, *tense*, *degree* and *mood*, being concatenated into a single feature vector. Which attributes are applicable depends on the base POS. The result is a fine-grained tagset of more than 700 tags in total. Table 2.2 shows an example of the POS attribute vectors. Each attribute is separated by a dot and the first attribute is the base POS category. A complete description of the tags used in the TIGER corpus can be found in Crysmann et al. (2005).

Table 2.2.: An example for a tagged German sentence by the RFTagger. Tag labels are described by Crysmann et al. (2005).

word	tag
farblose	ADJA.Pos.Nom.Pl.Fem
grüne	ADJA.Pos.Nom.Pl.Fem
Ideen	N.Reg.Nom.Pl.Fem
schlafen	VFIN.Full.3.Pl.Pres.Ind
wutentbrannt	ADJD.Pos
.	SYM.Pun.Sent

To use a relatively large number of tags might improve discrimination abilities regarding above mentioned disambiguation purposes. Including morphosyntactic information could help to improve word agreement, especially for morphologically rich languages as German.

But concerning POS tagging, the problem of data sparsity arises when using a large tagset. The possibility of encountering tag combinations during testing without being observed during training increases. Schmid and Laws (2008) argue that modeling the context probability with a simple *n*-gram approach despite utilizing appropriate smoothing techniques is not sufficient.

As main innovation, they propose to decompose the context probability into a product of attribute probabilities and then use decision trees to model the latter.

This way the problem of data sparsity is alleviated. Moreover, the consideration of a larger context is possible. The context of up to 10 preceding words were used instead of typically only 2 for trigrams.

An example of this decomposition is shown in Equations 2.29 – 2.35 for the tag sequence corresponding to the sentence “das blaue haus” (see Figure 2.6). For illustration purposes the number of attributes and the context length are vastly reduced. The context probability $p\left(t_i \mid t_{i-k}^{i-1}\right)$ is first defined on whole attribute vectors (Eq. 2.29). The leading numbers (1:,2:, later 0:) correspond to the predecessor (1), pre-predecessor (2) and current word (0). The attribute vectors are then split up in single attributes (Eq. 2.30 – 2.32). Each attribute is modeled to be still dependent on its base POS category but independent from other attributes. The formula is then approximated using only those conditional attributes which are necessary to decide for the current attribute value (exemplary in Eq. 2.33 – 2.35).

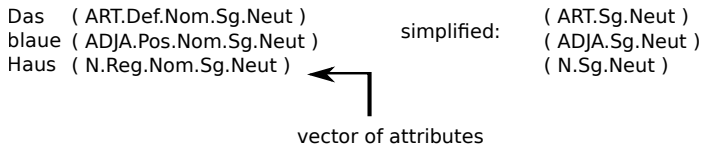


Figure 2.6.: The tagged German utterance “Das blaue Haus”. Tag labels are described by Crysmann et al., 2005.

$$p(t_i|t_{i-k}^{i-1}) = p(\text{N.Sg.Neut} \mid \text{2:ART.Sg.Neut, 1:ADJA.Sg.Neut}) \tag{2.29}$$
$$= p(\text{N} \mid \text{2:ART, 2:ART.Sg, 2:ART.Neut, 1:ADJA, 1:ADJA.Sg, 1:ADJA.Neut}) \tag{2.30}$$
$$* p(\text{N.Sg} \mid \text{2:ART, 2:ART.Sg, 2:ART.Neut, 1:ADJA, 1:ADJA.Sg, 1:ADJA.Neut, 0:N}) \tag{2.31}$$
$$* p(\text{N.Neut} \mid \text{2:ART, 2:ART.Sg, 2:ART.Neut, 1:ADJA, 1:ADJA.Sg, 1:ADJA.Neut, 0:N, 0:N.Sg}) \tag{2.32}$$
$$\approx p(\text{N} \mid \text{2:ART, 1:ADJA}) \tag{2.33}$$
$$* p(\text{N.Sg} \mid \text{1:ADJA.Sg , 0:N}) \tag{2.34}$$
$$* p(\text{N.Neut} \mid \text{1:ADJA.Neut, 0:N}) \tag{2.35}$$

Which attributes are relevant is modeled by a decision tree.

The questions are chosen during training by optimizing an information gain criterion. Pruning of the tree shall prevent overfitting to the training data (Schmid and Laws, 2008). In addition, smoothing is necessary to avoid zero probabilities either due to data sparseness or because of ungrammatical input during testing.

A tree is built for each value of each attribute of the feature vectors given a context of predefined length. In a leaf the context probability of the attribute value the tree was built for, given the context defined by the path to that leaf is stored. An exemplary tree that models the probabilities of the current tag being a noun in nominative case (N.nom) given different contexts is depicted in Figure 2.7.

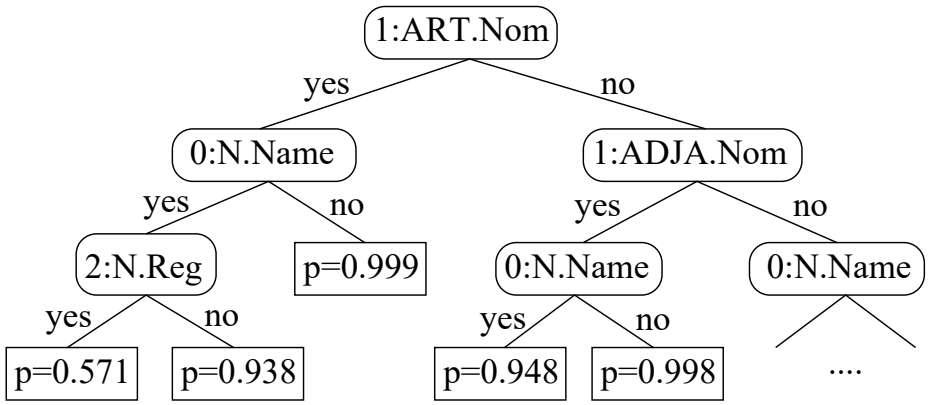


Figure 2.7.: An exemplary decision tree to model the probability of the current tag being a noun in nominative case (N.nom) given, by Schmid and Laws (2008).

The RFTagger is able to estimate lexical probabilities for unknown words that were not seen during training. It utilizes strategies like regarding capitalization and word endings. The latter are arranged in a suffix tree for efficiency. Statistics of known words are used to anticipate the lexical probabilities of an unknown word with the same ending.

Schmid and Laws (2008) report accuracies up to 91.07% for predicting German fine-grained POS tags when trained and tested on the TIGER corpus. As the number of fine-grained tags exceeds most other POS tagging approaches, direct comparison is difficult. When only predicting the smaller set of STTS base POS tags, the tagger achieves state-of-the-art accuracies of up to 97.97%. Best results could be obtained with a context width of 10 preceding words. However, the highest gains are achieved when switching from 2 to 3 context words and performance improvements seem to be almost converged at a context width of 5.

2.2.2. Lemmas

Utilizing the lemma information has been proposed to reduce data sparseness (Sennrich and Haddow, 2016; Sennrich et al., 2009).

A lemma thereby is a linguistic term describing the citation form of a set of words (Collins English Dictionary, 2017; Sennrich and Haddow, 2016). All inflectional variants of a word share the same lemma. Verbs are usually represented by their infinitive and nouns in the singular nominative case. For example, the lemma of *go*, *goes*, *going*, *went*, and *gone* is *go* (Collins English Dictionary, 2017). The specific choice of the representative of each equivalence class that shares a common lemma is in the context of this work of no concern, as the systems do not work on character level. It is only necessary that class labels are distinguishable.

In this work we use the TreeTagger in order to derive the lemmas of the German target words (Schmid, 1994b, 1995).

2.2.3. Dependency Labels

Dependency grammars are a grammar formalism reflecting the syntactic structure of a sentence (Jurafsky and Martin, 2014). In this formalism each word in a sentence is dependent on a single other word (its syntactic head). This binary relation can be expressed by an arc, resulting in a tree structure as depicted in Figure 2.8. The verb is usually taken as the structural center resembling the root of the tree and, therefore, does not have a syntactic head. The dependency relations can be interpreted with the grammatical function a word fulfills in relation to its head. In Figure 2.8 we can see that the noun "I" is *subject* of the verb "prefer" whereas the noun "flight" is a *direct object*. These grammatical functions are referred to as *Dependency Labels*.

Because the entire dependency tree cannot be easily expressed on word level, the dependency labels are used instead.

To extract dependency labels for the German target language we use *ParZu* (Sennrich et al., 2013). It was first described by Sennrich et al. (2009) and builds on the same architecture as the English *Pro3Gres* parser (Schneider et al., 2008). ParZu is described as a hybrid dependency parser build upon external tools for POS tagging and morphological analysis, and merges a manually crafted rule-based grammar with a statistical disambiguation component that is trained on the TüBa-D/Z treebank (Telljohann et al., 2004).

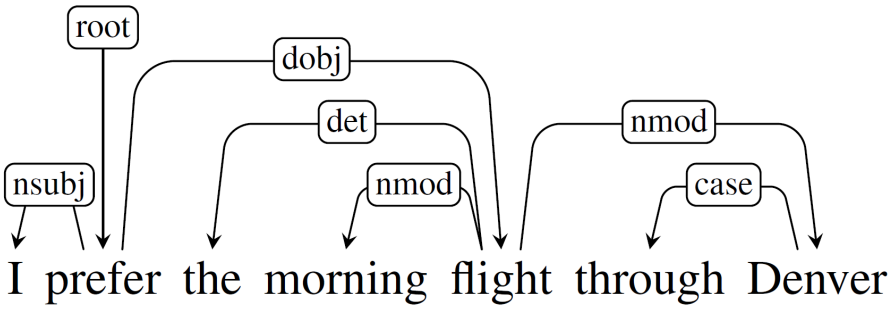


Figure 2.8.: Illustration of a dependency-style analysis using a standard graphical method by Jurafsky and Martin (2014)

2.2.4. IOB Tags

Sennrich and Haddow (2016) suggest annotating sub-word units with tags that encode the sub-word structure. Therefore IOB tags are proposed. In their approach **B** marks whether a sub-word unit forms the beginning of a word, **I** stands for inside and **E** flags the end of a word. The **O**-tag is used for units that cover an entire word. Advantages of IOB tags include the possibility to disambiguate the different types of sub-word units. Otherwise end units cannot be distinguished from whole words, as well as begin units from units inside a word. Also the BPE segmentation approach used in this work does not enable the NMT system to be aware of the mapping of sub-words to words. For example it is not prohibited by design that due to failing generalization efforts the network can generate incomplete words[3]. Thereby when additionally providing the IOB structural information we assume the NMT system to more robustly learn that words comprised of several sub-word units have to be finalized properly. The derivation of IOB tags once the data is segmented using BPE is straight foreward.

2.3. BLEU Evaluation Metric

The *bilingual evaluation understudy* (BLEU) metric proposed by Papineni et al. (2002) is the most widely used automatic evaluation metric in the field of machine translation (e.g., Han and Wong, 2016; Koehn, 2009; Liu et al., 2011; Song et al., 2013). *Automatic* evaluation as opposed to the assessment of a system's translation performance by human judges is necessary because of its availability, low cost, speed, reproducibility, language independence and objectivity[4] (Dorr et al., n.d.).

But as automatic evaluation algorithms are not able to rate quality criteria of translations, e.g., intelligibility, adequacy and fluency directly, they have to regress to low level similarity measures, comparing translation hypotheses generated by an MT system with high quality *reference translations* provided by human translators (Han and Wong, 2016).

[3] as negative side effect the incomplete word will then be merged with its successor word during post-processing

[4] Koehn (2009) suggests a possible bias of the BLEU metric towards phrase-based statistical MT which is similarly concerned with (the generation of) short phrases of high integrity

Thereby, a fundamental challenge consists in the lack of an unambiguous ground truth. When translating natural languages, usually many different valid translations exist to a given source sentence without an inherently objective ranking of quality. BLEU alleviates this problem by allowing multiple reference translations for a source sentence. The rationale behind the approach is that translation quality improves the closer a hypothesis is to the references and the aim of the evaluation metric is to quantify this "translation closeness" (Jurafsky and Martin, 2009).

The BLEU metric is thereby based on modified n-gram precisions. An n-gram is a sequence of n subsequent words and the concept of precision (and recall) is commonly used in information retrieval and in binary classification (Powers, 2007). An adoption of precision as a sentence-wise similarity measure between a hypothesis translation output generated by an MT system and reference translations can be defined on the basis of n-gram matching.

All n-grams of a certain order n found in any reference sentence comprise the set of *relevant* occurrences. The perspective of *precision* is to minimize the number of non-relevant n-gram predictions in the hypothesis translation, whereas completely hypothesizing all n-grams in the references is of no concern. On the other hand, the *recall* measure would demand for maximization of coverage, i.e. ideally completely hypothesizing all n-grams from all references—regardless of any non-relevant predictions which additionally may be made.

Instead of comparing each reference individually, they are considered collectively to allow phrasings from different references in different parts of the hypothesis.

On a sentence basis, n-gram precision might be defined as in Equation 2.36 (cf. Abele, 2014):

$$\text{precision}_n = \frac{\sum\limits_{n\text{-gram}\in H} \text{count}(n\text{-gram}, H) \max\limits_{R\in\mathfrak{R}}(\delta(n\text{-gram}, R))}{\sum\limits_{n\text{-gram}\in H} \text{count}(n\text{-gram}, H)}\,, \tag{2.36}$$

where $\text{count}(n\text{-gram}, H)$ is the number of occurrences of a specific n-gram in the hypothesis string H. $\mathfrak{R}$ is the set of references, the δ function checks whether the n-gram occurs in a reference R and the max operator ensures that it is sufficient for an n-gram to appear in any reference in order to be accounted.

An attempt to define sentence level n-gram recall in this scenario using a slightly different intuition is approached in Equation 2.37.

$$\text{recall}_n = \frac{\sum\limits_{n\text{-gram}\in H} \min(\text{count}(n\text{-gram}, H), \sum\limits_{R\in\mathfrak{R}} \text{count}(n\text{-gram}, R))}{\sum\limits_{R\in\mathfrak{R}} \sum\limits_{n\text{-gram}\in R} \text{count}(n\text{-gram}, R)}\,. \tag{2.37}$$

Recall is not used in BLEU, as with multiple references the requirement of a hypothesis to cover *all* phrasings of *all* references at once is not reasonable. In above definition of recall the numerator slightly differs from the definition of precision and takes into account that not more occurrences of a specific n-gram should be hypothesized than are backed up by the references taken all together. It could be used in the definition of precision as well. Which one to choose is of no concern as the authors of BLEU take this thought even one step further, when introducing the *modified* n-gram precision. Here the n-gram counts of

the hypothesis to be accredited are even further restricted by the maximum occurrence in any *single* hypothesis. The complete proposed formula for modified *n*-gram precision already appropriately averaged over the entire test set $\mathfrak{T}$ renders to

$$p_n := \text{precision}_n^{\text{mod}}(\mathfrak{T}) = \frac{\sum\limits_{(H,\mathfrak{R})\in\mathfrak{T}} \sum\limits_{n\text{-gram}\in H} \text{count}_{\text{clipped}}(n\text{-gram}, H, \mathfrak{R})}{\sum\limits_{(H,\mathfrak{R})\in\mathfrak{T}} \sum\limits_{n\text{-gram}\in H} \text{count}(n\text{-gram}, H)} , \tag{2.38}$$

where

$$\text{count}_{\text{clipped}}(n\text{-gram}, H, \mathfrak{R}) = \min\left(\text{count}(n\text{-gram}, H), \max_{R\in\mathfrak{R}} (\text{count}(n\text{-gram}, R))\right) \tag{2.39}$$

(cf. Papineni et al., 2002). The clipping of counts penalizes the over-generation of words in the hypothesis.

For the final BLEU metric the modified precisions of various orders n are combined. According to the authors shorter *n*-grams shall account for correct word choice (adequacy), longer *n*-grams for word order (fluency). As the probability to match longer *n*-grams decays exponentially, the precisions are logarithmized and possibly weighted. To compensate for not explicitly modeling recall and as precision is biased towards shorter hypotheses, a special term called *brevity penalty* BP is introduced to penalize hypotheses that are too short:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} . \tag{2.40}$$

Then,

$$\textsc{Bleu} = \text{BP} \cdot exp\left(\sum_{n=1}^{N} w_n \log p_n\right) , \tag{2.41}$$

where r is the effective reference corpus length[5], c is the total length of all hypothesis sentences and N the maximal order of *n*-grams to be considered.

As usually $N = 4$ and uniform weights of $1/N$ are recommended, this further simplifies to

$$\textsc{Bleu} = \text{BP} \times \left(\prod_{n=1}^{4} p_n\right)^{\frac{1}{4}} , \tag{2.42}$$

the geometric mean of the modified *n*-gram precisions weighted by the brevity penalty (Song et al., 2013).

Advantages and shortcomings of the BLEU metric found in the literature (e.g., Koehn, 2009; Liu et al., 2011; Song et al., 2013) include that, on one hand, the metric is simple, fast, easy to implement, language and domain independent, does not require additional resources nor task specific adaptation and, most important, highly correlates to human judgment. On the other hand, researchers criticize that BLEU is not representative at sentence level (scores rather have to be averaged over a large test set), that it does not account for similar phrasings or synonyms nor for an importance ranking of words and it

[5]for each hypothesis the reference closest in length is used to add up to the effective reference corpus length

does not explicitly include the recall measure. Also, absolute BLEU score values gained by different researchers on different experiments are not easily compared directly as they depend on the data, domain, language pair and number of reference translations deployed (Koehn, 2009).

3. Related Work

This chapter presents work that previously utilize word factorization approaches in machine translation. They were at first successfully used in Statistical Machine Translation (SMT). Therefore, a brief presentation will be given. Following this, we outline related work regarding factors in Neural Machine Translation (NMT). We thereby distinguish between approaches that are concerned with integrating factors on source language side of an NMT system as to those dealing with the target language side.

3.1. Factors in Machine Translation

Statistical machine translation has been the state-of-the-art approach to machine translation for over two decades. Including linguistic knowledge has already proven to be helpful to the field of machine translation during this period. Therefore, we give reference in order to acknowledge these ambitions. However, it must be kept in mind that the principles of SMT and NMT differ substantially, albeit the essential rationales behind the procedures remain related. Because this work does not focus on SMT in general, no insights about the essential principles are given, the interested reader is referred to Koehn (2009). Accordingly subsequent subchapters address the factorization approaches recently issued for the field of neural machine translation.

3.1.1. Statistical Machine Translation (SMT)

Koehn et al. (2007,2006) introduced the basic concept of using extra information on word level to the field of machine translation. The authors generalize the phrase-based statistical machine translation (PB-SMT) approach by substituting each word in surface form with a vector of factors of this word. Factors used for experiments include lemma, POS, morphological information and statistically derived word classes additionally to the surface words.

The translation is broken down into different steps as shown in Figure 3.1 (based on Koehn and Hoang (2007)). The analysis step refers to the mapping of a word to its additional factors derived by external tools. Translation steps are then carried out for each factor independently. Independent translation is necessary to not further increase the data sparseness problem (Birch et al., 2007). Thereby, the same basic phrase-based modeling approach is used for each factor. Lastly, a generation step uses a statistical model that operates at target word level[1] to map combinations of target factors to possible target words.

[1] i.e., no context is taken into account

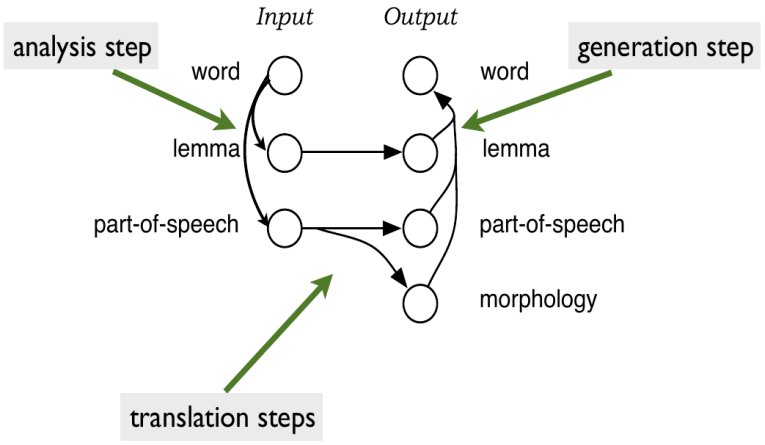


Figure 3.1.: Translation steps for SMT by Koehn and Hoang (2007).

Different scenarios were proposed. In the first scenario solely the *additional* factors are translated and then the target word is generated from their respective translations on target side, as shown in Figure 3.1. The source words themselves are not translated directly. In SMT, this procedure is helpful, as it allows morphological variants of source words to be mapped to their common lemma in order to share statistics, hence alleviating data sparseness problems. In order to be able to generate the correct morphology on target side, morphology information is used as an additional factor and translated independently. As another advantage, inflectional variants may be generated on target side, that are not included in the bilingual training corpus[2]. From the perspective of Neural Machine Translation these issues are still relevant but attenuated by the strong learning capabilities of the neural network, which allow the representations of similar words, including morphological variants, to relate. On the other hand, these learning capabilities are strong enough to learn from different tasks in parallel, therefore shifting the idea from *individual* translation to jointly present the additional data to the network, allowing it to find useful insights in the data itself. We further assume the utilization of a crude model for generation of surface word forms on target side to undermine the strong generation capabilities of the NMT model.

A second scenario for factored SMT proposes to use the additional information on target language side to build dedicated language models that operate on the respective additional factor exclusively. For example, as part of speech tags are less manifold than surface words, traditional n-gram language models of higher order could be used. In the currently most prominent NMT formulation however no explicit language model is deployed, as the decoder is basically a target language model already. Nevertheless, we assume the underlying motivation to achieve improvements in grammatical integrity, like word order or word agreement to be the same, although again, NMT models are stronger in these aspects already in advance (Bentivogli et al., 2016).

[2]as the word generation model on target side might be trained with additional monolingual data

3.1.2. Neural Machine Translation (NMT)

Regarding neural machine translation, first experiments with additional knowledge at word level were reported for the source language only (Hoang et al., 2016; Sennrich and Haddow, 2016). Experiments regarding the target language were performed since end of 2016 (Garcia-Martinez et al., 2016a,b; Nadejde et al., 2017).

As the experiments performed in this thesis are based on the English to German translation task, we focus on the comparison of results regarding this translation direction.

3.1.2.1. Factorization of Source Language

Sennrich and Haddow (2016) proposed the first approach to integrate factors into an NMT system. Their work focuses on the factorization of the source language words only. The target language remains unchanged. Again, external tools are used to automatically derive a conceptually arbitrary amount of word level factors for the source words.

A challenge consists in achieving compatibility of those factors gained at word level with the sub-word modeling approach used. The authors propose to simply copy the word level factors for all sub-word units of the same surface word. In the encoder's input layer, a separate vocabulary and embedding matrix is used for each factor. In each encoder time step the respective embedding vectors of all factors of the current input are then concatenated to form the overall embedding vector which is ultimately used as input to the system. The further procedure of encoding and decoding does not differ from a baseline attentional NMT system as described in Section 2.1. We will denote this procedure *joint encoding*.

In their experiments, the factors used are the BPE sub-word segments, IOB tags, lemmas, POS tags, morphological information and dependency labels. They report a significant improvement of up to +0.8 Bleu for English to German translation when using all factors except the morphological information. Even slightly better results are obtained for this translation direction when only using POS tags as additional factor.

Hoang et al. (2016) conducted similar experiments. As main innovation, they propose to use distinct attention mechanisms for each source factor in combination with the commonly used single attention mechanism. For English to German translation, improvements of up to +0.5 Bleu are reported. However, improvements could not be shown for all experiments consistently.

Details on design choices will be further elaborated in Section 4.3.

3.1.2.2. Factors on Target Side

Garcia-Martinez et al. (2016a,b) were the first to examine the factorization of target words. They proposed to replace the target words by a combination of lemma and POS together with morphosyntactic features. The system is trained to generate those factors as unit of translation instead of words. After decoding has come to an end, an external component is used to obtain a target word sequence from the sequence of additional target factors. This procedure is conceptually related to the work of Koehn and Hoang (2007) described in Section 3.1.1. Therefore, the reasons and advantages are the same as stated above. The external tool that generates the target words' surface forms from the additional factors is

not bound to the limitations of the bilingual corpus and therefore may generate unseen morphological variants and thus could not be predicted by the NMT system itself. The authors suggest this procedure to effectively increase the amount of possible output words for morphologically rich target languages while decreasing the target vocabulary size of the NMT system, as morphological variants do not have to be covered there. This motivation is less relevant to this thesis as we use the BPE sub-word encoding to achieve open-vocabulary translation. However, it is uncertain whether a linguistically more sophisticated approach could outperform the crude BPE segmentation technique.

Unfortunately, no significant performance improvements over a baseline system could be reported. Nevertheless, some interesting experiments were carried out, as for example, different alternatives to combine the factor embeddings, different ways to feed back the last decoder output as input to the next decoding step, and the idea to introduce an artificial dependency between lemma and factors to be generated.

Regarding this work, the main disadvantage is the requirement on an external tool which generates the full word forms on target side from the NMT output and has therefore to be dedicated to the chosen factors. This makes the approach less generic. We advise the avoidance of any such severe post processing strategies as they undermine the powerful end-to-end generation capabilities of the NMT system.

Nadejde et al. (2017) introduce tags based on the *combinatory categorial grammar* (CCG) formalism by Steedman (2000) as additional syntactical information to NMT. CCG tags have been successfully used by Birch et al. (2007) in a factored SMT system similar to the proposal by Koehn and Hoang (2007) referenced in Section 3.1.1. The main advantages of CCG tags is their ability to capture information about the structure of the sentence at word level. They were dubbed as an "approach to almost parsing" because they leave only minor ambiguities compared to a complete parse tree of the sentence's syntactic structure (Bangalore and Joshi, 1999).

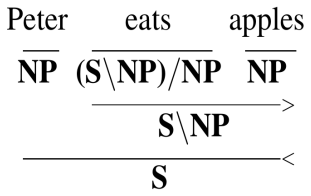


Figure 3.2.: Simple CCG example by Birch et al., 2007.

Figure 3.2 shows an annotated example sentence. *Peter* and *apples* are noun phrases (NP) and the verb *eats* is the root of the sentence (S), indicating the presence of a noun phrase on the left (\NP) and another one on the right (/NP). Where exactly in the sentence the latter are located is not defined by the grammar. The final word level CCG tags are NP, (S\NP)/NP and NP.

Certainly the NMT system cannot interpret the information about the sentence structure encoded in the tags *directly*. But when there is enough training data, it may deduce this knowledge by the observation of how tags co-occur with other tokens in certain situations. The authors argue that a tag at mid sentence might function as a reminder for decisions based on long range dependencies where words at the end of the sentence depend on

words at the beginning. When basing its decision on a CCG tag at mid sentence, the system is released from the burden to remember this information on the whole span of the sentence. The authors argue that especially coordination and prepositional phrase (PP) attachments can be generated more accurately.

They performed different experiments with CCG tags for English both on source and on target side. On source side, the performance reported by Sennrich and Haddow (2016) as stated above could not be outperformed using CCG tags.

For the target language factorization, three architectures were proposed, that are discussed in more detail in Chapter 4. For the first approach, dubbed as *serializing*, the target factor values are shown to the decoder in an alternating order during training, sharing the same decoders' input and output layers among all factors. When using CCG tags for English as target language translating from German improvements of +0.6 Bleu are reported. The English to German translation direction was not investigated.

The two remaining approaches are based upon multi tasking architectures (Luong et al., 2015) and both could not lead to translation performance improvements when used with target factors.

Also, the factorization of *both*, source and target side at once, were investigated. Using dependency labels for German as source and CCG tags for English as target language an improvement of up to +1.0 Bleu could be achieved.

The main disadvantage of the CCG grammar is that, at the time of writing, the availability of openly accessible parsers is unfortunately limited to only a few languages including English. But for the German target language as used in this work no such parser could be found. Hence no experiments were conducted using CCG tags.

4. Target Factors for Neural Machine Translation

This work aims to probe the usefulness of integrating additional knowledge into an NMT system by means of factorizing the target language words. After stating fundamental assumptions, we briefly highlight the different design decisions in oder to give a comprehensive overview of the chosen approaches. Regarding the combinatorial explosion of possible design options, we have to focus on reasonable choices. We therefore first carry out a basic n-best list re-ranking approach as a preliminary study. We then present the idea to alternate target factors, dubbed as serialization approach. Next, we propose to jointly decode target factors. And lastly, we combine source and target factorization. The usefulness of the proposed approaches are then empirically evaluated in Chapter 5.

4.1. Assumptions

This work applies the following assumptions:

1. Factor information can be encoded at word level.
2. Factors are generically interchangeable.
3. The (sub-)word factor has to be generated by the translation system.

These assumptions enable us to tightly integrate additional information about the target language into an NMT system. We thereby seek for a generic solution that is not tailored to a specific type of information or would lead to architectural changes dependent on a specific factor. Moreover it is important that factors can be represented at word level to use them in conjunction with the (sub-)word units during training, as opposed to training different tasks with entirely different training data (e.g., multi-tasking of translation and parsing by Luong et al. (2015), Niehues and Cho (2017)). Lastly, we want the NMT system to output the word factors (in our case sub-word units) itself. We do not want to utilize another additional tool that generates words from a variety of additional factors, as this would lead to a dependency on factor selection. Also, we believe the strong learning and generation capabilities of the neural network architecture to be most appropriate for the task of target word generation and therefore do not want to resort to more primitive tools or interfere the end-to-end generation process.

4.2. Levels of Component Sharing

A major design decision driving the construction of systems that integrate target factors is the sharing of components. When adopting the basic attentional encoder-decoder architecture to operate on multiple target factors, we first consider how much of the system to be dedicated to the different factors and which parts to share among all factors.

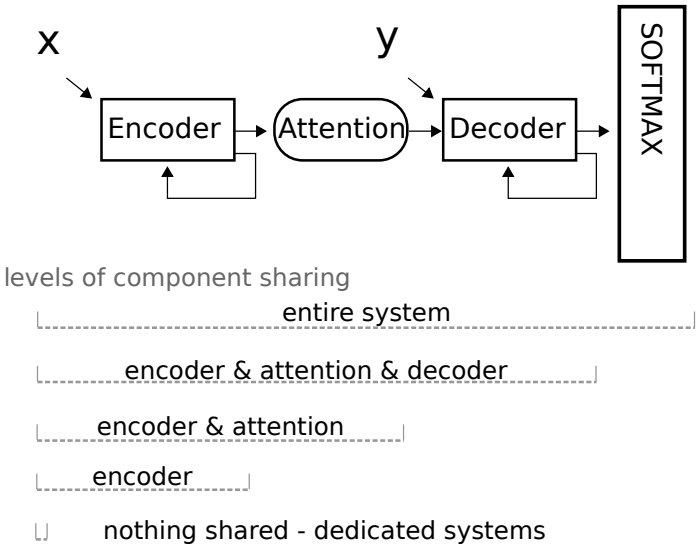


Figure 4.1.: Levels of component sharing (extended version based on Niehues and Cho, 2017)

Entire Architecture is Shared It is possible to use the entire architecture without any changes. In order to integrate target factors they can be interleaved with the stream of target words, called serializing and will be described in greater detail in Section 4.3.2. It is the so far only approach known from literature, that could successfully be used for target factorization in NMT. Another possibility would be to use a concatenation of all target factors into a single factor. But this strategy is not helpful, as it would even increase data sparseness and it is not feasible, as the vocabulary of the concatenated factors would be too large to meet the memory restrictions of modern GPUs.

Shared Decoder In this level of sharing, only the softmax output layer is distinct for each target factor. Especially the same decoder is used to predict all target factors synchronously. We call this a *joint decoding* and go into greater detail in Section 4.3.2.

Shared Encoder w/o sharing of Attention Mechanism Also, solely the encoder can be shared while utilizing different decoders, presumably one for each target factor. Thereby, either common or distinct attention mechanisms are used. As a main disadvantage of this approach, no information is shared on the target side of the system. Whatever shared knowledge is to be learned among target factors has to be propagated back into the encoder, which by design is rather concerned with the source language. We therefore did

not further investigate this approach. It could however be shown that this approach is useful to multi-task additional *source* information (Luong et al., 2015; Nadejde et al., 2017; Niehues and Cho, 2017). For the target language, so far no improvements could be shown (Nadejde et al., 2017).

Dedicated Translation Systems Lastly, it is possible to not make use of any system sharing, by building a dedicated translation system for each target factor. These systems might then be used in a n-best list re-ranking approach as described in the next section. As an advantage, all factors may make use of a full blown NMT system. A disadvantage includes the missing opportunity to share common knowledge among factors within a shared system.

4.3. Proposed Approach

The following subsections explain our design decisions leading to three approaches on how to integrate target factors into an NMT system. Additionally, we will also include source factors and state in our approach.

4.3.1. N-Best List Re-Ranking

First, we propose a n-best re-ranking approach. It is meant as a preliminary study to possibly—but not necessarily—show the usefulness of target factor information. Any sound solution should be able to surpass the results gained with n-best list re-ranking.

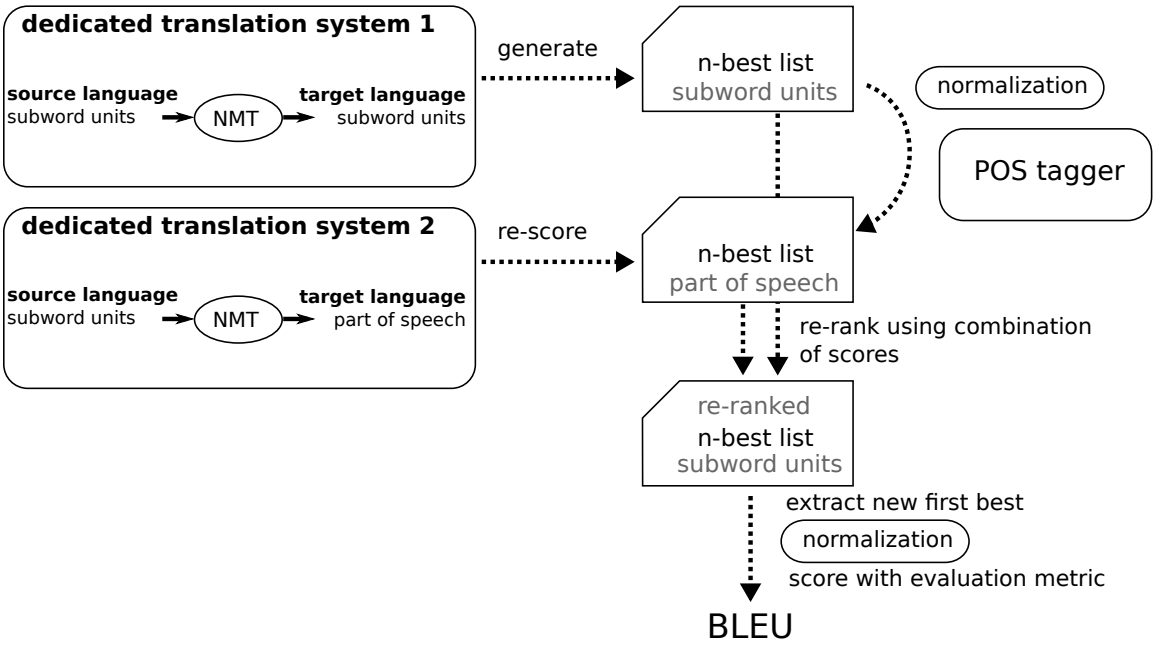


Figure 4.2.: N-Best List Re-Ranking

The procedure which is depicted in Figure 4.2 is as follows. First, two dedicated translation systems are trained. The first of which is in our case a baseline system that translates

sub-word units in a source language to sub-word units in a target language. It is used to obtain a n-best list of reasonable size. A n-best list literally covers the *n* best translation hypotheses to each input source sentence in the test set. They are generated by the beam search procedure of the baseline system. The basic assumption behind every n-best re-ranking approach is that the n-best list contains hypotheses that are of better quality than the top ranked choices. Finding those alternatives could lead to better overall performance. Therefore, a second system is used to score each pair of input source sentence and target sentence hypothesis from the n-best list a second time, called re-scoring. Thereby, the second system can be designed to be more specialized to the re-scoring task, as it does not have to find translation hypotheses itself but rather re-score existing ones. Finally, the scores of both systems are combined to overall scores, e.g., by taking the sum. The latter is used, as scores are logarithmized probabilities and the sum of scores therefore corresponds to the joint probability given by the product of probabilities. The ordering of hypotheses in the n-best with respect to the overall scores is assumed to change, ideally revealing new top ranked choices of better translation quality. The new first best choices in the n-best list are evaluated with an evaluation metric, in our case the Bleu metric.

The re-scoring system we use in this proposal is a dedicated translation system, that maps source sub-word units to one of the additional target factors in isolation, e.g., part of speech in Figure 4.2. Note that the dedicated system does not take sub-word units of the target language into account. For training of such a system, the training corpus' target side is tagged with the factor information and then this information is solely used as target sequence for training. The system architecture does not have to be altered. Because NMT is a rather generic sequence-to-sequence mapping approach, this procedure is feasible.

In the re-ranking scenario, this implies that the target sentence hypotheses in a n-best list generated by the baseline system are tagged by the external tool used to gain the respective information, e.g., POS tags. Therefore, sub-word units are normalized to words in surface form and some minor post-processing is performed. After re-scoring and re-ranking the original hypotheses are restored in the new order and the top-ranked choices are evaluated. Utilizing the tagger not only during training, but also at translation time is a major difference to the remaining proposals, where the latter is prohibited.

We use this rather intricate approach of the dedicated translation system, instead of simply deploying a tag based target language model for re-scoring purposes, because any more elaborate approach will also be confronted with the challenge to generate target factor information based on a source sentence.

If the re-ranking approach was successful, we assume it to immediately prove the usefulness of the target factor information in a translation scenario. However, it is unclear how much potential improvements the original n-best list offers, e.g., the possibility remains that it does not contain any better translation choices than its first best proposals. Therefore, this study will not *necessarily* show the usefulness of target factors.

4.3.2. Serialization of Target Factors

As the second approach, we propose to use a single feature stream containing all target factor tokens in an alternating order. The concept is depicted in Figure 4.3, where target words are interleaved with their respective POS and dependency labels. Parallel to the

preparatory phase of this work, Nadejde et al. (2017) published a similar proposal. We adopt their naming convention to call this a *serialization* approach.

As an advantage, no architectural changes are necessary. Only the respective target factor vocabularies have to be concatenated to a joint vocabulary. Also, the target sentences of the bilingual training corpus are annotated with their respective additional factor information and these sequences are then interleaved to form a single target sequence of target factor tokens. As the NMT system is confronted with a serialized stream of factor tokens, it learns to generate them during translation likewise. A factor dependent special character sequence was added to each token in advance in order to easily distinguish the different target factors in a translation hypothesis after decoding. For the final translation hypothesis, all additional factor tokens are discarded. However, we also evaluate them independently to investigate the overall prediction capabilities of the NMT system.

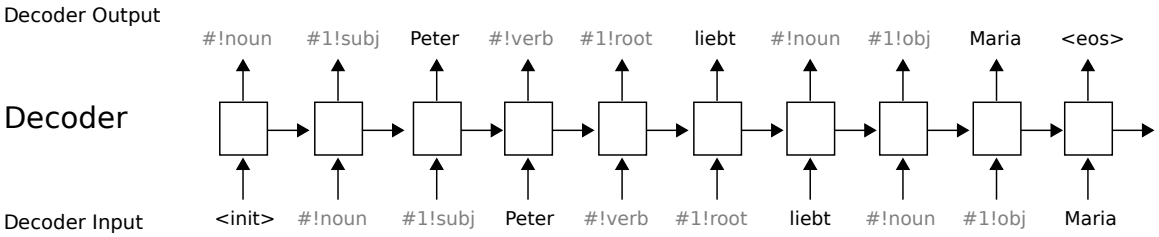


Figure 4.3.: Serialization of POS, dependency labels and target words for German target sentence “Peter liebt Maria” (Peter loves Mary). Simplified illustration omitting input from the encoder and the complexity of softmax output layers. Additional factors are prefixed with special character sequences in order to separate them easily in the translation hypothesis.

Summarizing the related work section on current proposals that deal with NMT target factorization directly, we find that the serialization approach is so far the *only* proposal that could show translation quality improvements, at least when using CCG tags. Therefore, we validate these findings on a different experimentation setting. We are specifically interested in whether improvements are reproducible with factors other than CCG tags. Further experimentation includes the order of factors and the word-level to sub-word level mapping strategy, as described as follows.

Order of Factors When serializing all factors into a single stream, we have to define an order in which factors appear in this stream. First of all, it is important to note that we want to interleave factors at word level, i.e., all factors assigned to the first word are to appear in the final stream first, followed by the factors of the second word, etc. We are then especially interested in the question whether the additional factors should precede or succeed their related (sub-)word unit. In case of using the additional factors first, from a probabilistic viewpoint, the prediction of the (sub-)word unit thereby will be conditioned on the latter. This might enable the neural network to first choose a broader category, e.g., the appropriate morphosyntactic information and then use this information when deciding on the correct surface word form of the subsequent (sub-)word unit. However, due to the deployment of gated recurrent units it might as well ignore the additional

information. We also include experiments were the additional factor tokens follow their respective (sub-)word unit. Because the shared knowledge that might be gained must have been learned during training, the precise order might be irrelevant, which could be indicated by these experiments.

Word-Level to Sub-word Level Mapping Strategy In this work, BPE units are used as main word factor at sub-word level, while most additional factors are generated at word level. IOB tags are an exceptional case, as they reflect the sub-word structure and are therefore defined on sub-word level. Thus, it is necessary to establish a mapping strategy between word and sub-word level. For the serializing approach, we use three different procedures. Firstly, the additional factor tokens might be combined with all their respective sub-word units belonging to the same surface word only a single time. I.e., preceding or succeeding the first sub-word unit, depending on the chosen order, but not appearing together with the remaining sub-word units of a segmented word. Thereby, we loosen the strict alteration pattern. Secondly, in order to maintain the strict pattern of succession, it is possible to copy a word level factor to all its related sub-word units. Lastly, it is possible to concatenate the tokens from an additional factor vocabulary with IOB tags to establish sub-word level mapping. We assume this strategy to increase data sparseness. Also, this strategy is not feasible if the size of the factor vocabulary grow too large, as in case of lemmas.

Expected Advantages and Disadvantages We assume possible advantages of the proposed method to include the following aspects. All factor tokens are projected into a common embedding space instead of subspaces of limited dimension. Therefore, each projection may make use of the expressiveness of the overall embedding space. Especially, the embedding dimensionality of the most important target factor—the sub-word units—does not have to be reduced.

Also, we assume the system to learn to represent the dependencies among factors in a common space similar as it does for words as presented in Section 2.1.2. For example relations between as POS tag and its corresponding words might be expressed. As this implies that more information is modeled within the same space, the expressiveness of each factor might be effectively reduced, but it is left to the system to make the decision on how much bandwidth is used for each factor and how to possibly suppress redundancy. Moreover, illegal factor token combinations (e.g., the word “house” interleaved with the POS “verb”), which might occur during translation due to failing generalization efforts, at least do not affect the embedding layer – as opposed to the joint decoding approach presented in the next section.

Lastly, we assume the attention mechanism to be able to partially learn factor dependent attentional behavior. Although no dedicated attention mechanism is used for each factor, the former has to deal with only one distinct target factor token at each decoder time step.

As disadvantages, this approach is not able to model ambitious sub-word units with a different representation for each meaning, as the embedding vector for a sub-word unit cannot be chosen to differ for each of its possible meanings. However, due to its powerful context modeling capabilities, we assume the model to have no difficulties with disambiguation when the sub-word unit is preceded by appropriate additional information

that would enable the latter. A rather application-oriented disadvantage includes that the target sentences double in length. Consequently, training and translation is more time and memory consuming (Nadejde et al., 2017). Also, due to vocabulary concatenation, the softmax output layer increases in size, leading to the same issues.

4.3.3. Joint Decoding of Target Factors

As the third approach, we propose to use a shared decoder with a separate softmax output layer for each target factor. The architecture is depicted in Figure 4.4. Encoder and attention mechanism are shared as well. We thereby aim to use a factorized representation also at the input layer of the decoder.

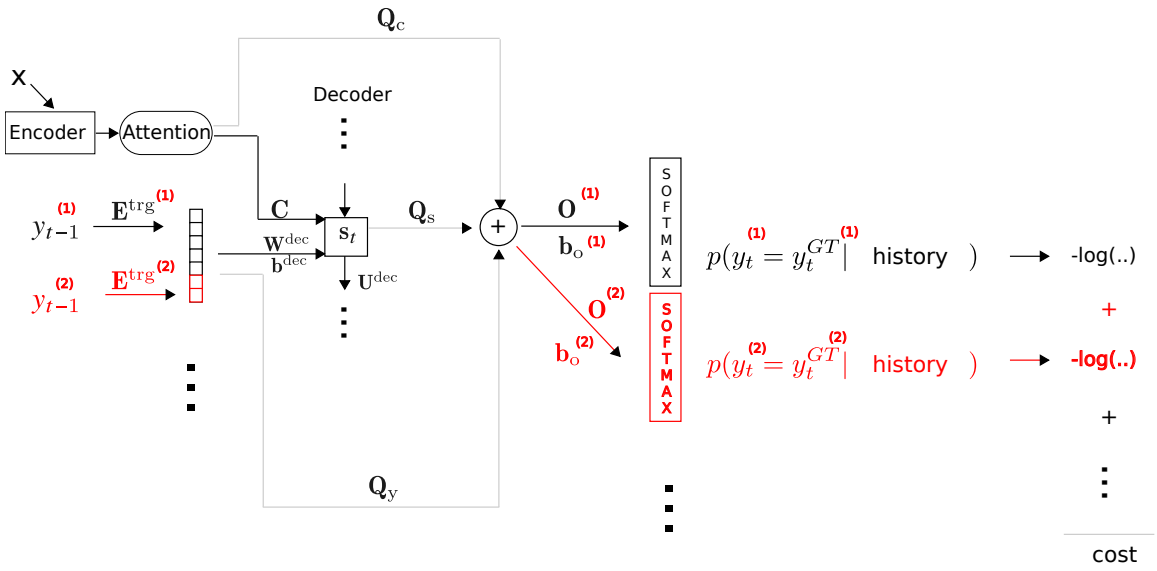


Figure 4.4.: Architecture for jointly decoding all target factors in parallel with a distinct softmax output layer for each. Also the decoders target input is factorized. (1) describes data structures for the sub-word unit base factor and (2) describes data structures for an additional factor. More additional factors can be introduced following the same scheme. Scores are added during training (possibly weighted). y_t^{GT} denotes the respective ground truth value. The figure is a modified excerpt from Figure 2.1.

Nadejde et al. (2017) use a similar approach except that they do not use a factorization of the decoder's input layer. They only use the distinct softmax output layers. This was also proposed among a series of multi-tasking approaches (e.g., Luong et al., 2015). When generating the next decoder output, their proposal does not take into account the additional factor information predicted by the system in the previous decoding step. Therefore, the system's generation of (sub-)word units cannot be conditioned on the *choice* of the additional target factors in the previous decoding step. Also, when only the (sub-)word units are used for decoder input, no dedicated representations for such homographs that are disambiguated by additional factors are possible. On the other hand, Garcia-Martinez et al. (2016a) use mostly the same architecture as presented here, but do

not predict the target word factor directly. Also, they do not utilize word segmentation. None of the above stated proposals were successful when used for target factors.

Integration of Embeddings A first design decision to be made is how to build the target-sided input of the decoder. Usually, the last decoded word is fed back to form the latter. We now want to be able to deal with multiple target factors that were predicted in the last decoding step. A common scheme in the related works concerned with factorization approaches in neural machine translation and neural language modeling is to use a dedicated embedding matrix for each factor and then concatenate the respective factor dependent embedding vectors to perform an overall embedding vector, as rendered in Equation 4.1 (cf. Sennrich and Haddow, 2016).

$$e^{\text{dec}}\left((y_{t-1}^{(1)},..,y_{t-1}^{(|F|)})\right) = \Big\|_{i=1}^{|F|} \mathbf{E}^{(i)} h^{(i)}(y_{t-1}^{(i)}) \tag{4.1}$$

Thereby, F is the set of target factors, $(y_{t-1}^{(1)},..,y_{t-1}^{(|F|)})$ is the vector of target factor tokens the search strategy decided to be the output of the previous decoding step, $\mathbf{E}^{(i)}$ and $h^{(i)}$ are factor dependent embedding matrices and one-hot functions, respectively, and $\|$ denotes vector concatenation. The final embedding vector $e^{\text{dec}}(\cdot)$ is then fed to the decoder.

Training Criterion In order to predict all target factors in parallel, the training criterion has to be slightly altered to take into account, the errors made by the forward propagation for the additional target factors as well. In order to compute the cost for a single sentence pair $(\mathbf{y},\mathbf{x})$ from the training corpus we decide to simply sum the (possibly weighted) negative log scores over all target words in the sentence pair.

$$\text{cost}(\mathbf{y}|\mathbf{x}) = \sum_{t=1}^{L_y}\sum_{i=1}^{|F|} -\lambda_i \log(P(y_t^{(i)}|c_t(\mathbf{x}),H)) \tag{4.2}$$

Thereby, λ_i are factor dependent weights, F is the set of target factors, L_y is the length of the target sentence and H are all so far predicted target factors $(y_1^{(1)},..,y_1^{(|F|)},..,y_{t-1}^{(1)},..,y_{t-1}^{(|F|)})$. The probability $P(y_t^{(i)}|\cdot)$ is predicted by the respective factor-dependent softmax output layers. Costs are then averaged over all sentences in a mini-batch. The remaining error backpropagation approach is not altered.

Beam Search When using all target factors that were decoded in the previous time step as input for the current decoding step, the beam search strategy has to be adapted. As target factors are decoded independently, we now face the situation that there is a beam for each target factor. In order to merge the beams, it is necessary that for each decoding time step every hypothesis of target factor A has to be combined with every hypothesis of target factor B and so on. This implies to build the cross product space of all possible target factor hypotheses at a given decoding time step (Garcia-Martinez et al., 2016a). As the cross product space of all factor vocabularies possibly exceeds the memory limitations posed by modern GPUs, careful design is necessary. A possible solution is to first truncate

the probability distributions of each target factor after the k most probable entries, where k is the beam width of the beam search strategy. As all factors are treated independently, applying pruning by the beam width first is feasible. This allows to build the cross product space where all factor hypotheses scores of the current decoding timestep are combined to overall scores. The remainder of the search strategy remains the same. A sentence is finalized when for the main (sub-)word unit factor the *<eos>* token is decoded, regardless of the hypotheses of the additional factors at that time step.

4.3.4. Combination With Source Factorization

Finally, we are interested whether the combination of our target factorization approaches with the integration of source factors is beneficial. For source factorization, we follow two approaches:

Firstly, we use the approach proposed by Sennrich and Haddow (2016). In this approach, a dedicated embedding matrix is used for each source factor. The respective source factor embedding vectors are then concatenated to form the overall input to the encoder, similar to Equation 4.1. We call this *joint encoding* as all source factors are presented to the encoder in parallel.

Secondly, we find that the serialization approach described in Section 4.3.2 could be transferred to the source language as well. As to the best of our knowledge, this idea has not been investigated so far for the source language.

5. Evaluation

In this chapter, we provide empirical evidence to the question whether the integration of additional knowledge into an NMT system by factorizing the target language can improve translation performance.

First, we describe the experimental setup, including the Neural Machine Translation Toolkit, the choice of data and hyper-parameters, the evaluation method and the baseline system. In the second part of this chapter, the results of the different approaches described in Section 4.3 are discussed.

5.1. Experimental Setup

All experiments were conducted translating from English as source language to German as target language. Hence, target factors were applied for the German language in approach 1 – 3. Additionally, in approach 4, factors were utilized first on source side solely and then on source and target side combined.

5.1.1. Neural Machine Translation Toolkit

The Nematus[1] NMT system by Sennrich et al. (2017) was used as a basis for all experiments. It is an implementation of the attentional encoder-decoder architecture described in Section 2.1.1. It is powered by the Theano Framework, that makes GPU implementation transparent, is capable of automatically computing symbolic differentiation and provides many other tools that are useful for various machine learning applications (Al-Rfou et al., 2016).

5.1.2. Data and Hyper-parameters

Training and Test Data In this work, training and evaluation is performed on data derived from TED[2] talks as used in the IWSLT evaluation campaign (Cettolo et al., 2015), publicly accessible at WIT[3] archive (Web Inventory of Transcribed and Translated Talks (Cettolo et al., 2012))[3]. TED (*Technology, Entertainment, Design*) is a non-profit organization that hosts conferences where guest speakers are invited to give talks that are made publicly available on the Internet free of charge. The talks were subtitled and translated by volunteers. Therefore, the translations are mostly of reasonable quality, but not always as sophisticated as if carried out by professional translators. The domain of spoken language usually is

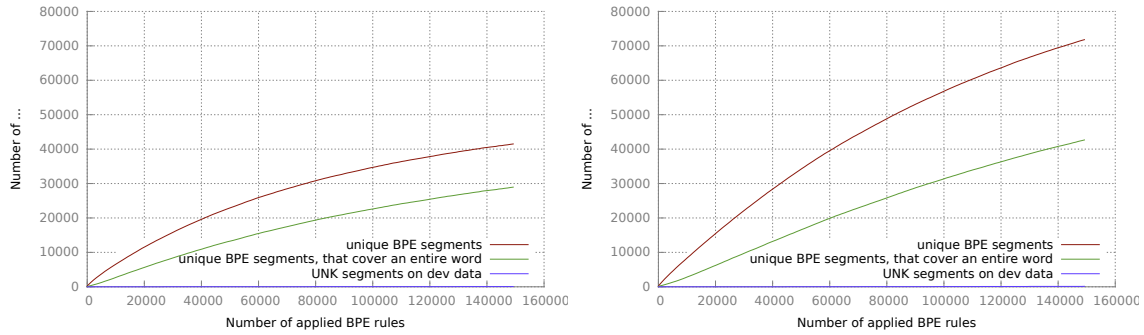
[1] `https://github.com/EdinburghNLP/nematus`
[2] `ted.com`
[3] `wit3.fbk.eu`

Table 5.1.: Size of Training Data

	English	German
Number of sentence pairs	189 thousand	
Number of tokens[4] in total	3.5 million	3.3 million
Number of unique tokens	51 thousand	111 thousand
Number of unique BPE segments	40 thousand	40 thousand
Number of BPE segments in total	3.6 million	3.6 million
Average sentence length in:		
tokens	18.6	17.3
BPE segments	18.9	19.1

concerned with spontaneous speech phenomena, like hesitations, false starts, repetitions, bad grammar, etc. However, due to intense rehearsal these talks show only a decent amount of such spontaneous speech phenomena (Cettolo et al., 2015). Nonetheless, compared to written language, a more vivid and pictographic style is used, containing many idioms, colloquialisms and metaphors.

Quantitative numbers describing the data are shown in Table 5.1 and Table 5.2. The number of sentence pairs is rather small compared to current approaches in the literature which use several million sentence pairs (e.g. Sennrich et al., 2015a). The experiments can therefore be framed as low resource setting. An advantage is that training converges in days instead of weeks. As German is a morphologically rich language, the German word vocabulary is more than twice as large as its English counterpart.



(a) en (b) de

Figure 5.1.: BPE segmentation: resulting vocabulary sizes, number of words that are not splitted and number of resulting UNK tokens on development data (test2013) as a function of BPE merge rules

We use BPE segmentation for vocabulary reduction as the presented in Section 2.1.7. To learn the merge rules we use a data basis consisting of the concatenation of source and target training data together with a large background corpus. We thereby assume the robustness of rule extraction to increase. We use about 60K merge rules for the German

[4]A token is a full word, punctuation mark, number or any other contiguous (special) character sequence.

Table 5.2.: Development and Test Data

	number of sentence pairs	number of total UNK tokens[5]		number of total UNK BPE segments	
		en	de	en	de
test2013	993	203	518	85	28
test2010	1565	210	783	71	65
test2011	1433	237	653	104	77
test2012	1700	267	756	124	41
test2014	1305	275	665	85	53
test set	6003	989	2857	384	236

language resulting in 40k sub-word units. Concerning joint BPE for English the same amount of rules would lead to a vocabulary comprising only 26k sub-word units. We decided not to use joint BPE but a different number of 135k rules for English leading to a 40k sub-word unit vocabulary as well, as this showed slightly better results for our baseline system.

Figure 5.1 plots statistics concerning the BPE segmentation as a function of learned merge rules. We can see that for German according to its larger vocabulary size also more BPE segments result when applying the same amount of rules. The relation between generated segments and learned rules is not linear, firstly because we use the above mentioned concatenation of various training data a basis for rule extraction. This implies that rules are learned that cannot be applied to a certain language and are instead simply ignored. Secondly, throughout rule application shorter segments are completely covered by longer segments and therefore do not occur in the resulting vocabulary anymore. Additionally we can observe that for German more than half and for English about two thirds of the resulting BPE segments cover an entire word. In case of convergence of the BPE rule extraction algorithm the vocabulary based on BPE segments would coincide with the non-segmented word vocabulary.

Table 5.2 shows statistics about our development and test data. Thereby “test2013” is used as development data during training and the remaining entries (test2010–12, test2014) comprise our test set. We can see that the number of unknown (UNK) tokens is significantly reduced when switching from word vocabularies to vocabularies based on BPE segmented sub-word units.

Target Factor Vocabularies The vocabularies were built upon the annotated training data. Differences to the original vocabularies used by the respective annotation tool are possible but irrelevant for the training procedure. This is because the NMT system can only learn about tokens that actually appear in the data. Any additional tokens simply cannot be

[5] no segmentation

learned and therefore the vocabularies may be truncated to contain only occurrences from the training data.

RFTags are parts of speech concatenated with other morphosyntactic features (number, case, gender, etc.) as descibed in Section 2.2.1.

The number of unique lemmas surpasses the number of unique sub-word units, because the former are not segmented. As can be seen from Table 5.3, the number of unique lemmas is however less than half as big as the number of unique word level tokens (111 thousand).

One would assume the number of unique lemmas to be smaller than what is displayed in Table 5.3 because especially for a morphologically rich language like German, many morphological variants of a word should map to a single lemma. Among others, a reason to the large number of lemmas is that a large number of singleton words map each to a unique lemma. There are about 58 thousand unique word tokens with a single appearance in the data. Instead of only deleting lemmas that occur only once, we take this a step further. The main reason to utilize lemma information is that words that share a common base form shall profit from each other, e.g., by being mapped to adjacent points in the embedding space. From this perspective, lemmas targeted only by a single morphological variant are useless, even if the lemma appears often in the data. Therefore, additional experiments which use a *reduced* lemma vocabulary, where at least two different words map to the same lemma were conducted.

Table 5.3.: Vocabulary sizes: number of unique tokens in German factor vocabularies. All vocabularies additionally contain *UNK* and *<eos>* tokens.

Factor	**Vocabulary Size**
BPE sub-word Units	40008
RFTags	723
Lemmas	50681
Lemmas (reduced)	16294
Dependency Labels	32
IOB Tags	4

Hyper-parameters The following hyper-parameters were used for all experiments.

As it is not feasible to search the vast space of parameter combinations for optimal choices, either standard parameters recommended by the creators of the Nematus system (Sennrich et al., 2017) or otherwise proposed in literature were used.

We use word embedding layers for source and target language each of the size of 500 dimensions and the encoders and decoders hidden layers comprise of 1024 units. As our dataset is much smaller than those used by the latter authors we assume these dimensions to be not chosen too small. An answer to the question whether they are chosen too big is given by Britz et al. (2017). The authors could show through extensive experimentations that too large dimensions do not harm performance. When using oversized hidden layers,

this might lead training not to converge properly—a behavior which was not observed in any of our experiments. The error backpropagation training is performed with the Adadelta algorithm (Zeiler, 2012), a variant of stochastic gradient descent with adaptive learning rate (Sennrich et al., 2017). The norm of the gradient is clipped to 1.0 (Pascanu et al., 2013).

We use a mini-batch size of 80 and for each mini-batch, sentences of similar length are used in order to speed up training (Sutskever et al., 2014). Also, the training data is shuffled with each training epoch, which refers to one complete training loop over the entire training data corpus. Sentence pairs where both sentences exceed 50 words are discarded from the training data, which is already taken into account in Table 5.1. This was done in advance in order to have the same data basis for all experiments—as some experiments have an effect on sentence length.

During translation, a beam size of 12 is used for all experiments and also for validation on a development set during training.

The early stopping criterion is used, supported by the Nematus system (Sennrich et al., 2017). The patience is set to 10.

5.1.3. Evaluation Method and Baseline

The results are presented in Bleu (metric described in Section 2.3). Evaluating NMT systems is not an easy task. In this section, we highlight the existing difficulties and the strategies taken to face them. We thereby also present our baseline system.

Training of a neural network not necessarily results in finding the global optimum. In order to not always be confronted with the same (possibly weak) local optimum, training of a neural network is designed to be non-deterministic. The non-deterministic behavior manifests itself in the random initialization of weights, random shuffling of data, probabilistic dropout and rounding errors. Consequently, when several copies of a system sharing the same hyper-parameters are trained independently, a different local optimum might be reached each time, resulting in different system quality, which makes comparison of different approaches not trivial.

Moreover, it is important to establish fair comparison among different systems. Due to convergence at different paces, we do not use a fixed amount of training iterations or training time for each system, but deploy the early stopping criterion. We thereby assume each system to be reasonably converged, allowing for fair comparison.

Exemplarily, we show the convergence behavior of our baseline system, which was run three times independently in Figure 5.2. For each baseline run, the scores on the development set are plotted as a function of training iterations. We can observe that all three baseline runs converge reasonably similar. Note that due to slight variations, the early stopping criterion is met at different iterations.

During the training of a system it is evaluated every 10k iterations and models are saved every 30k iterations. For evaluation, two strategies are pursued. First, the model that achieved the highest validation Bleu score during training is used for evaluation—dubbed as *single best* in all result tables. Secondly, *ensemble* decoding is performed on the four best models according to validation scores. In ensemble decoding, multiple models are used for translation. At each decoding step, the respective probability distribution for the

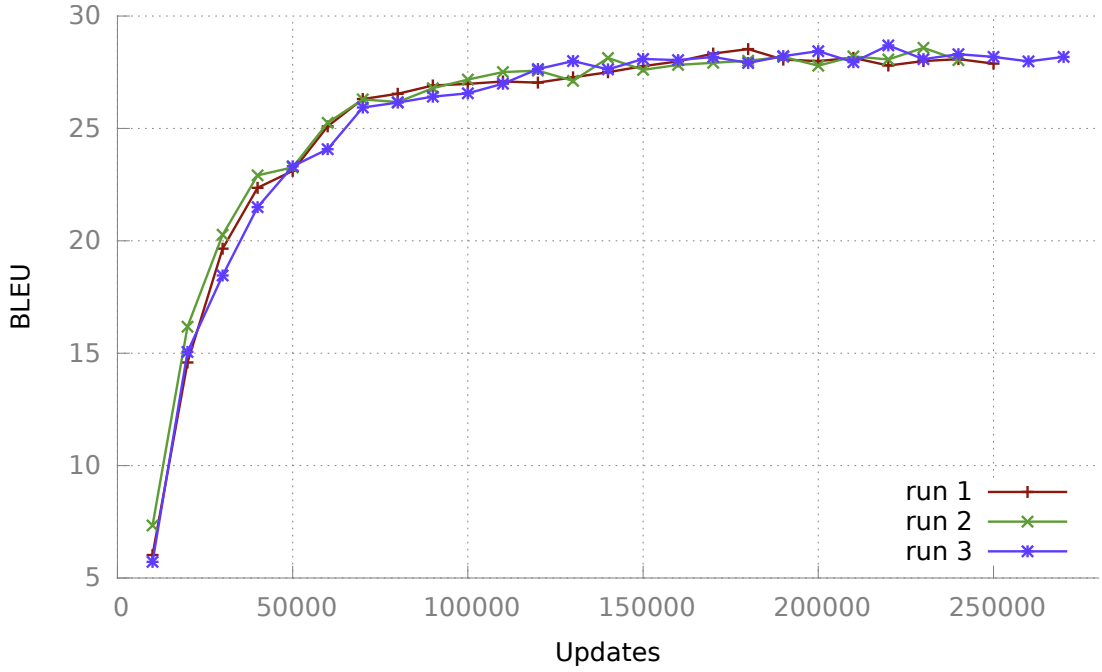


Figure 5.2.: Plot of validation BLEU scores on the development set as a function of training iterations.

next word choice is derived by each model in the ensemble. These probability distributions are geometrically averaged in order to obtain an overall distribution which is then further processed as if a single model was used (Sennrich et al., 2017). Sennrich and Haddow (2016) state that this procedure smooths the variance between single models.

Table 5.4 shows the BLEU scores for the three baseline runs on the development set and test set. We can see that the evaluation tasks seem to be of varying degrees of difficulty for the models to solve, with scores ranging between 24 and 30 BLEU. More importantly, we observe that the maximum deviation of single best scores between the baseline runs is considerable—it is listed in the final row. For *test2011*, a maximum difference of 0.79 BLEU is observed. This is of the same magnitude we assume possible improvements to be, due to target factorization. The ensemble scores show less deviation, possibly allowing for better comparison of different models. Also, we notice the ensemble scores to be significantly higher than the scores achieved with single best models. Therefore, we will give more priority to the ensemble scores, as this is the designated way of decoding—at least if not suffering from severe resource restrictions. However, we also report results achieved with the single best model, as ensemble decoding is a heuristic and might not fit all scenarios the same way (e.g., the number of models to be taken into account might differ if a system converges faster, etc.).

To further improve comparability, we average the results obtained for the individual test data files. Thereby, we observe a notable reduction of variance both for the single best and ensemble decoding. In Table 5.4, the maximum difference between the baseline runs is only 0.09 BLEU and 0.15 BLEU for single best and ensemble decoding, respectively. We will thus focus on the averaged values.

Table 5.4.: BLEU scores for the three Baseline runs on the Development and Test Set

Baseline	Dev Set		Test Set								Average Test Set	
	2013		2010		2011		2012		2014			
	single best	ens	single best	ens	single best	ens	single best	ens	single best	ens	single best	ens
run 1	28.53	28.81	26.71	**27.90**	**29.53**	30.11	25.65	26.74	24.19	24.78	**26.52**	27.38
run 2	28.59	28.96	**27.11**	27.72	28.74	**30.43**	25.79	**27.05**	**24.36**	**24.93**	26.50	**27.53**
run 3	**28.69**	**29.10**	26.59	27.56	28.96	30.28	**25.97**	26.91	24.20	24.90	26.43	27.41
$\Delta(\max, \min)$	0.16	0.29	0.52	0.34	0.79	0.32	0.32	0.31	0.17	0.15	0.09	0.15

5.2. Experiments and Results

Limited resources force us to evaluate only a limited amount of experiments, therefore a selection of reasonable approaches was motivated in Chapter 4.

Despite the usage of modern GPUs, evaluation is a time-consuming task. Therefore, only a selection of experiments were possible.

5.2.1. N-Best List Re-ranking

As described in Section 4.3.1 for n-best list re-ranking, at least two systems are necessary. A baseline system is deployed to generate a n-best list. For re-scoring, dedicated translation systems are used that translate English sub-word units to German RFTags and Lemmas, respectively.

Word-segment to Tag Translation First, let us examine the performance these systems can achieve when deployed for their dedicated translation task. Here, we also report on systems translating to dependency labels and IOB tags which were not used for re-ranking.

The results in Table 5.5 show that the performance of the baseline systems operating on sub-word units can be clearly surpassed by all three systems. Thereby, systems translating to RFTags and lemmas achieve about +4 BLEU above the baseline for ensemble decoding, when averaged over the test set. For dependency labels, this difference is even higher, about +17 BLEU.

This is not surprising as the vocabularies of the respective tags are either much smaller or as for lemmas we expect that the most common words map to a reduced number of lemmas. Therefore on average, the system is confronted with a smaller number of translation choices, thus the translation task is easier.

However, the results are inferior to what a tagger could achieve when directly deployed on the target language. That is because the task of translating sub-word units from one language to tags in another language is more challenging than expected (Garcia-Martinez et al., 2016a).

Table 5.5.: BLEU scores for dedicated word-to-tag-translation systems. They are used for re-scoring.

Dedicated Systems	Dev Set		Test Set								Average Test Set	
	2013		2010		2011		2012		2014			
	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
RFTags	32.60	33.80	30.48	32.34	31.77	33.94	29.61	31.08	28.48	29.20	30.09	31.64
Lemmas	32.37	33.23	30.50	31.13	32.80	33.83	30.11	30.84	28.57	29.13	30.50	31.23
Dependency Lables	46.35	46.57	44.88	45.74	44.86	45.75	42.81	43.27	43.28	43.43	43.96	44.55
IOB	69.64	56.54	76.83	60.58	77.14	66.69	76.00	65.44	72.67	58.50	75.66	62.80

Note that the sub-word unit to lemma translating system might show the potential that could be exploited by a system that is able to entirely predict the correct morphology in the target language[6]. Also note that when evaluating these systems, we cannot rely on human annotations of the test set as references, but have to resort to deploying the respective taggers. Therefore, results might be slightly biased which is not problematic, because these systems are used only indirectly as re-ranking system or as means of comparison for the tag generating capabilities of the systems in experiments described in subsequent sections.

N-Best List Re-ranking Table 5.6 shows the results achieved with the re-ranking approach. We use the baseline system (run 1) to write the n-Best list of 100 target hypotheses for each input source sentence[7]. Thereby, consequently a beam size of 100 is used during translation which leads to slightly better results for most test cases (detailed results for all test cases can be found in Table A.1). As re-ranking strategy the sum of original scores and re-scored pendants are used. For re-scoring, systems translating to RFTags and Lemmas are deployed. Because it is often already helpful to use a second instance of the same system for re-scoring, we also use run 2 and 3 of the baseline system to ensure a fair comparison. We can see that all re-ranking approaches could lead to better results than baseline (run 1). Also, we show that the results were not obtained by chance. To demonstrate chance level, we randomly choose hypothesis from the n-best list and calculate mean and variance of scores several such random re-rankings achieve. We can see, that all of the proposed approaches significantly surpass chance level. Moreover, by also choosing the worst hypotheses from the n-best list we show the large range of quality the latter covers.

[6] including some side effects as translating names and numbers, etc. correctly which are mapped to common lemmas, respectively.

[7] because baseline run 1 was the first to be available. We take into account that run 2 achieved slightly better results by using it as a re-scoring system

Table 5.6.: BLEU scores for Re-Ranking

Re-Ranking	Average Test Set	
	single best	ensemble
baseline (run 1. beam 12)	26.52	27.38
baseline (run 1. beam 100)	26.88	27.43
RFTags	27.05	27.77
RFTags. dims reduced	27.03	27.74
Lemmas	27.37	27.72
RFTags+Lemmas	27.61	**28.03**
baseline (run 2)	27.42	27.73
baseline (run 3)	27.55	27.86
baseline (run2 + run3)	**27.81**	27.98
Average Random	23.84	24.61
Variance	0.03	0.06
WORST	17.23	19.21

When using the combination of scores produced by systems translating to RFTags and lemmas, we can achieve an improvement of +0.6 BLEU over baseline for ensemble decoding. This shows that the information learned by these systems is at least useful. However, compared to re-ranking the baseline with its two other independent runs the same improvements can be achieved. This implies that at least for the re-ranking scenario, no unique information is introduced, that could not have been already gained by the data on (sub-)word level itself.

We also experiment with a system translating to RFTags that uses reduced word embedding and decoder hidden layer sizes of 80 and 512, respectively. This experiment was conducted because in subsequent sections, experiments are performed where (sub-)word unit and RFTag prediction are generated synchronously by a single system which will reduce the systems capacity for each task. For re-ranking purposes, the dimension reduction seem not to harm performance.

Lastly, we want to answer the question if the scores from the re-ranking systems alone are helpful as well. These are given in Table 5.7. The combination of RFTags and lemmas still slightly outperforms the original baseline. But utilizing the two other baseline runs is more helpful this time, as their expressiveness is much higher, allowing them to be used without the original scores.

5.2.2. Serialization of Target Factors

In this experiment, we examine the serialization approach described in Section 4.3.2. Table 5.8 shows the results in terms of BLEU of our main translation task, i.e., when scoring against references based on words. We only display the average test set results. A

Table 5.7.: Bleu scores when using re-score values only

	Average Test Set	
	Rescore only	
	single best	ensemble
RFTags	26.25	26.52
RFTags. dims reduced	26.19	26.46
Lemmas	26.79	27.12
RFTags+Lemmas	27.26	27.56
baseline (run 2)	27.19	27.44
baseline (run 3)	27.33	27.50
baseline (run2 + run3)	**27.63**	**27.82**

breakdown by the single test files and also results for the additional factors however can be found in the appendix.

In Table 5.8, "X » Y" implies that for each target word, factor X precedes factor Y in the serialized stream. In order to evaluate whether the order of inserting morphological information before sub-word units ("BPE") or afterwards has a relevant impact on translation quality, both directions were examined in experiments 2.1 – 2.7 regarding RFTags and lemmas ("LEM"). Also different strategies of mapping word level factors to sub-word level are compared. The respective column is dubbed as "SWM" for sub-word mapping strategy. Thereby, "single" means that for words comprising multiple sub-word units the respective additional tags are used only before (or after, depending on the order) the first sub-word unit. Accordingly, "copy" means that the additional tags are copied for each sub-word unit. "IOB" implies that the IOB tags are already defined on sub-word level.

Additionally, dependency labels ("DEP") and IOB tags were tested. Lastly, we investigated reduced lemma vocabularies ("LEM (red)"), described in Section 5.1.2. For better comparability, we first state our strongest baseline described in the last section.

We can observe that most systems (2.1 – 2.11) slightly outperform the baseline system for ensemble decoding, but mostly only by about +0.1 to +0.4 Bleu which might as well be due to random fluctuations. Only system 2.9, based on the reduced lemmas, seems to clearly outperform the baseline for ensemble decoding by +0.7 Bleu. Though this result is not fully transferable to the single best evaluation method. Here, only an improvement of +0.13 Bleu could be achieved. For single best decoding, none of these systems (2.1 – 2.11) can improve over baseline by more than +0.35 Bleu.

The results are inconclusive also concerning the order of succession. We assumed the precedence of additional factors before sub-word units to be beneficial. Notably, the results are independent of whether the factor information precedes or succeeds the sub-word units.

Moreover, it hardly can be stated, which word level to sub-word mapping strategy succeeds. While for lemmas, "single" outperformed "copy", the reversed case seems to

Table 5.8.: BLEU scores for main translation task (prediction of target words), approach Serializing

	Serializing: WORDS		Average Test Set	
sys-id		SWM	single best	ensemble
	baseline (run 2)		26.50	27.53
2.1	RFTags » BPE	single	26.82	27.68
2.2	RFTags » BPE	copy	26.19	27.22
2.3	BPE » RFTags	single	26.48	27.73
2.4	BPE » RFTags	copy	26.55	27.71
2.5	LEM » BPE	single	26.34	27.76
2.6	LEM » BPE	copy	25.85	27.35
2.7	BPE » LEM	copy	26.16	27.62
2.8	LEM (red) » BPE	single	26.48	27.69
2.9	LEM (red) » BPE	copy	26.63	**28.24**
2.10	DEP » BPE	single	26.68	27.82
2.11	IOB » BPE	single	26.68	27.92
2.12	DUM » BPE	single	26.85	27.94
2.13	DUM » BPE	copy	**27.06**	27.73

apply for reduced lemmas. Regarding RFTags, "single" seems to be beneficial at least, when the RFTags precede subword units.

We are also interested in whether the prediction of the additional factors can benefit from the sub-word information. We therefore score those predictions against factor based references gained by annotating the word based references with the respective external tools. Results are stated in Table 5.9 to Table 5.12. We chose the dedicated translation systems from Section 5.2.1 as respective baselines to compare to. We can observe that only dependency labels and IOB tags seem to profit. The latter by up to +20 Bleu which is not surprising, as we assume the sub-word structure annotation task to be highly coupled with sub-word units themselves.

Lastly, we are interested in whether the improvements gained are truly due to the additional information introduced or could be explained by the increased capacity of the architecture that might be gained, because the serialization effectively increases the depth of the encoder. Therefore, we conducted a comparison experiment (2.13), where single tags without special meaning is interleaved with the sub-word unit stream (dummy tag, "DUM"). The results show that still improvements compared to baseline can be achieved, although no useful information is introduced. For ensemble decoding +0.2 Bleu were gained which is the scope of most other systems and could imply random fluctuations. For single best decoding however the best result of +0.5 Bleu in this series of experiments can be observed. Therefore either the random fluctuations are much higher than our experiment with three baseline system runs in Section 5.1.3 suggest or side-effects due to the architectural setup are introduced. Also the ensemble decoding results might simply more stable and reasonable as their single best decoding counterparts. Additionally, as they always lead to better translation performance in terms of Bleu, we shall give higher focus to the ensemble decoding results.

We also conducted the latter experiment with the sub-word mapping strategy "single" (experiment 2.12). We thereby result with a system that annotates the sub-word structure, as for words that are split to several sub-word units only the first unit is tagged. Assuming that at least the ensemble decoding results are reasonable we can see that the same results can be reached as when using the slightly more complex IOB tags to annotate subword structure (as in experiment 2.11).

Table 5.9.: Bleu scores for prediction of RFTags, approach Serializing

	Serializing: RFTags		Average Test Set	
sys-id		SWM	single best	ensemble
	RFTags (Baseline)		30.09	31.64
2.1	RFTags » BPE	single	**30.84**	**31.76**
2.2	RFTags » BPE	copy	29.40	30.51
2.3	BPE » RFTags	single	30.26	31.53
2.4	BPE » RFTags	copy	29.21	30.82

Table 5.10.: Bleu scores for prediction of Lemmas, approach Serializing

	Serializing: Lemmas		Average Test Set	
sys-id		SWM	single best	ensemble
	LEM (Baseline)		**30.50**	**31.23**
2.5	LEM » BPE	single	29.84	31.13
2.6	LEM » BPE	copy	29.19	30.42
2.7	BPE » LEM	copy	29.28	30.66

Table 5.11.: Bleu scores for prediction of Dependency Labels, approach Serializing

	Serializing: Dependency Labels		Average Test Set	
sys-id		SWM	single best	ensemble
	DEP (Baseline)		43.96	44.55
2.10	DEP » BPE	single	**44.60**	**45.21**

Table 5.12.: Bleu scores for prediction of IOB tags, approach Serializing

	Serializing: IOB		Average Test Set	
sys-id		SWM	single best	ensemble
	IOB (Baseline)		75.66	62.80
2.11	IOB » BPE	single	**82.95**	**82.84**

5.2.3. Joint Decoding of Target Factors

We examined the approach of joint decoding of target factors. Table 5.13 shows results averaged over our test set.

Table 5.13.: BLEU scores for main translation task (prediction of target words), approach Joint Decoding

	Joint Decoding: WORDS		Average Test Set	
sys-id		SWM	single best	ensemble
	baseline (run 2)		26.50	**27.53**
3.1	BPE | RFTags	copy	26.41	27.03
3.2	BPE | LEM	copy	25.65	26.72
3.3	BPE | DEP	copy	26.03	26.71
3.4	BPE | IOB	IOB	26.26	26.90
3.5	BPE | RFTags_IOB	IOB	25.99	26.89
3.6	BPE | RFTags | IOB | LEM	copy	25.08	25.52
3.7	BPE | RFTags | IOB | LEM (beam 100)	copy	25.29	
3.8	BPE | NUM	–	**26.54**	27.15
3.9	BPE | RFTags, (red. dec. dim. 1)	copy	26.26	26.94
3.10	BPE | RFTags, (red. dec. dim. 2)	copy	26.01	26.74
3.11	BPE | RFTags, (red. dec. dim. 3)	copy	26.35	26.86
3.12	BPE (emb 10) | RFTags | IOB | LEM	copy	24.67	25.33
3.13	BPE (emb 10) | RFTags | IOB | LEM, (TW)	copy	24.11	24.92
3.14	BPE (delayed +1) | RFTags | IOB | LEM	copy	21.89	22.83

First, we applied different factors individually. In the tables, "X | Y" stands for X and Y being jointly decoded. Note that the order of factors is irrelevant in this experiment. We used BPE sub-word units respectively with RFTags, Lemmas, Dependency Labels and IOB Tags. Moreover, we concatenated RFTags with IOB Tags ("RFTags_IOB"). This is not feasible with Lemmas, as this would result in a too large vocabulary. When specifying the dimensions of the dedicated factor embeddings, it is necessary that their sum does not exceed the embedding layer size of the baseline system (500 units) in order to avoid the introduction of additional parameters. Table 5.14 shows our choices which give high priority to the subword units, but are also based on the vocabulary sizes of the respective factors.

As clearly can be seen, unfortunately, none of our attempts could outperform the baseline system. Therefore, firstly, the question arises whether the implementation is flawed. To motivate the opposite, we designed an experiment where a simple counting task is multi-tasked in the joint decoder architecture together with translating BPE sub-word

Table 5.14.: Embedding dimensions chosen for additional target factors. A total size of 500 is used for every experiment. In each case the remaining dimensions are used for BPE sub-word units.

Embedding dimensions	
RFTags	20
LEM	167
DEP	20
IOB	5
RFT_IOB	20
NUM	3
RFT | IOB | LEM	20 | 5 | 175

units, named "NUM" in Table 5.13. The task is to repeatedly count from 0 to 2. Thereto, the target sentences of the training data are annotated with a "0 1 2 0 1 2..." sequence instead of linguistic tags. We assume this counting task to use only little of the networks capacity and to be completely useless information with regard to the sub-word generation task, while still comprising enough complexity to show that something reasonable could be learned. The results meet our expectations. The counting task can be learned nearly perfectly. Meanwhile it could be shown, that the BPE sub-word task was not influenced negatively and achieved a level at least comparable to baseline. This provides evidence, that our implementation is valid.

The next question to be investigated is whether the utilized beam size of 12 is too small for the search strategy that includes all factor combinations. Therefore, we repeat decoding of system 3.6 with a beam size of 100. We only consider single best decoding because the respective results already highlight, that although small improvements to using a beam size of 12 for the same system are achievable, the results are still substantially lower than baseline Bleu scores.

In order to make sure that the translation quality of the BPE segments did not deteriorate at the cost of improving the generation quality of the additional factors, we also examined performance in this direction (Table A.8 and following). We state, that also the prediction quality of the additional factors deteriorate for RFTags, Lemmas and Dependency Labels. The RFTag-Baseline outperforms all other approaches for every single test set. The difference to the nearest best results achieved by the joint decoding approach varies from 0,13 to 1,76 Bleu for RFTags. So there is no serious deterioration, but also no improvements could be achieved. The same situation was observed regarding Lemmas and Dependency Labels. Also in this scenarios the respective baseline systems achieved the best results regarding best single and ensemble for every single test set. The only exception could be seen with IOB Tags. But still the results mostly remain lower than the results achieved by the serializing approach.

We are then interested in whether the different factors are possibly modeled entirely independent by the system. In order to show that dependencies exist, we modified the

beam search strategy to randomly hypothesis the additional factors, while the sub-word unit prediction remains unchanged. That means that in the input layer of the decoder the previously predicted sub-word unit is combined with random choices for the additional factors. We could then investigate a drop of translation performance by about 4 Bleu which implies that the systems sub-word unit generation is influenced by the additional factors but not entirely deteriorates when confronted with wrong choices for the latter. This shows that factors are not treated independently by the system but suggests that dependencies are rather weak. Whether a knowledge sharing among factors is performed, remains unclear. However it seems from the results in Table 5.13 that results worsen the more information has to be modeled.

In the following, we describe several further experiments conducted, while unfortunately none of them could show any success as can be seen in the lower part of Table 5.13. Firstly, we were interested in whether we could force the system to share information about the factors by simply reducing the decoders hidden layer dimension. Maybe the model's capacity was chosen to large in first place, allowing the factors to be treated mostly independent. However, by reducing the decoders hidden layer size to 512, 384 and 256 units in experiments 3.9, 3.10 and 3.11, respectively, no improvements could be shown. Interestingly, performance does not decrease significantly either.

The next approach was to decrease the embedding dimension for the sub-word units (3.12 and 3.13). The motivation was that thereby information must mostly be used from the additional factors due to the decreased expressiveness of the former. Alexandrescu and Kirchhoff (2006) could achieve best results when not using the word factor. For systems 3.12 and 3.13, we use an embedding of size 10 for the sub-word units, and 30, 5, 455 for RFTags, IOB and Lemmas, respectively. In system 3.13, we also increased the training weight for sub-word units from being equally weighted to 0.8 (while using 0.1, 0.5 and 0.5 for RFTags, IOB and Lemmas, respectively).

Lastly, we were interested whether the results gained with the serialization approach are due to the succession of factors in contrast to their parallel application in joint decoding. Thus, we conducted an experiment where the sub-word units are padded in order to appear jointly with the additional factors that relate to the direct successor of the sub-word unit (dubbed as "delayed +1"). This attempt however shows worst results.

In summary, the assumption arises that the decoder is not able to extract the combined information if the tags are jointly presented.

5.2.4. Combination With Source Factorization

Lastly, we evaluate the combination of target factorization with source factorization. In order to achieve comparability, it is at first necessary to state results that can be gained when solely applying source factorization.

5.2.4.1. Source Factorization

First, Joint Encoding Source Factors was evaluated. Second, the Serialization approach, presented in Section 4.3.2 was utilized for Source Factorization.

For the English source language we use POS, lemmas and IOB as additional factors. POS and lemmas are derived with the TreeTagger (Schmid, 1994b, 1995), and the generation of IOB tags is straightforward. Note, that for the source language we use POS tags, because RFTags are not available for English.

Joint Encoding refers to the approach that all source factors are presented to the encoder in parallel and was proposed by Sennrich and Haddow (2016). The results are shown in Table 5.15. The factors' order is irrelevant in this scenario. The first experiments were conducted by combining sub-word units with POS tags, lemmas and IOB tags. The word to sub-word mapping strategy "copy" was applied for POS tags and lemmas as reported by Sennrich and Haddow (2016).

Best results could be achieved with POS tags, slightly outperforming baseline by +0.27 Bleu for single best and +0.09 Bleu for ensemble decoding. All remaining results are either close or worse than baseline. Especially they are worse than what we expected after reviewing respective literature. A reason could be that we possibly face a more challenging translation task.

Table 5.15.: Bleu scores for main translation task (prediction of target words), approach Joint Encoding

	Source Factored		Average Test Set	
sys-id		SWM	single best	ensemble
	Baseline (run 2)		26.50	27.53
4.1	BPE | POS	copy	**26.77**	**27.62**
4.2	BPE | LEM	copy	26.47	27.42
4.3	BPE | IOB	IOB	26.48	27.35
4.4	BPE | POS | LEM	copy	26.41	27.54
4.5	BPE | POS | LEM | IOB	copy	26.41	27.49

Serialization of Source Factors Table 5.16 shows results obtained when following our proposal to apply the serialization approach to the source language. Thereby, slightly better results are achieved than for joint encoding stated above. When using all available source factors, we can achieve improvements of +0.32 Bleu for single best and +0.24 Bleu for ensemble decoding.

While applying the word to sub-word level mapping strategy "single" with POS tags achieved better results compared to "copy", but the opposite case seem to be valid for lemmas. This could imply a dependency of the strategy on the factor type.

Again, we performed a special experiment to verify that possible improvements are based on the factor information and not introduced by the design of the architecture (experiment 4.13). Thereby, a single tag that cannot comprise useful information is interleaved with the

sub-word sequence (dummy tag, "DUM"). Results are at the same level as baseline, which substantiates that otherwise observed improvements are due the source factor information.

Table 5.16.: Bleu scores for main translation task (prediction of target words), approach Serialization of Source Factors

	Alternating		Average Test Set	
sys-id		SWM	single best	ensemble
	Baseline (run 2)		26.50	27.53
4.6	POS » BPE	single	**26.98**	27.72
4.7	POS » BPE	copy	26.47	27.51
4.8	LEM » BPE	single	26.20	27.36
4.9	LEM » BPE	copy	26.62	27.76
4.10	IOB » BPE	IOB	26.43	27.52
4.11	POS » LEM » BPE	single	26.52	27.71
4.12	POS » LEM » IOB » BPE	single	26.82	**27.77**
4.13	DUM » BPE	copy	26.73	27.54

Table 5.17.: Bleu scores for main translation task (prediction of target words), approach Combination of Source and Target Factorization

	Words				Average Test Set	
sys-id	Source		Target		single best	ensemble
	Baseline (run 2)				26.50	27.53
5.1	BPE | POS | LEM	copy	RFT » BPE	single	27.00	28.13
5.2	POS » LEM » BPE	single	RFT » BPE	single	**27.60**	**28.46**
5.3	BPE | POS | LEM	copy	BPE | RFT	copy	26.48	27.47
5.4	POS » LEM » BPE	single	BPE | RFT	copy	26.74	27.76
5.5	BPE | POS | LEM	copy	LEM » BPE	single	26.62	28.19

5.2.4.2. Combination of Source and Target Factor Approaches

In Table 5.17, we state our results for the main translation task (target word prediction). Thereby, we evaluate every combination of source/target serialization ("X » Y") with joint encoding/decoding ("X | Y"). As this experiment was designed prior to the availability of the other experiments' results, we decided to use POS tags and lemmas on source side and RFTags on target side together with the respective sub-word units.

The results for using the serialization approach on both sides clearly outperforms baseline by about +0.9 Bleu for ensemble decoding and even +1.1 Bleu for single best

Table 5.18.: BLEU scores for prediction of target RFTags, approach Combination of Source and Target Factorization

	RFTags				Average Test Set	
sys-id	Source		Target		single best	ensemble
	RFT (Baseline)				30.09	31.64
5.1	BPE | POS | LEM	copy	RFT » BPE	single	31.51	32.47
5.2	POS » LEM » BPE	single	RFT » BPE	single	**31.73**	**32.76**
5.3	BPE | POS | LEM	copy	BPE | RFT	copy	29.49	30.55
5.4	POS » LEM » BPE	single	BPE | RFT	copy	29.71	31.00

decoding. Interestingly, the improvements are higher than taking the sum of improvements from the respective experiments performed solely on source or target side. Moreover, also the quality of RFTag prediction increases, as can be seen in Table 5.18. This strengthens us in our assumption that shared knowledge was utilized, maybe even across the language barrier.

As expected from our findings in Section 5.2.3, systems with joint decoding achieve inferior results. However, the combination of target serialization with joint encoding of source factors could also outperform baseline.

5.3. Discussion

In order to summarize our findings, we display the best results that could be achieved for each respective experiment in Table 5.19. For a more detailed analysis we also state the results gained on the individual test set translation tasks. As expected the combination of source and target factorization outperforms all other approaches, at least with regard to ensemble decoding. It thereby achieves best results on all four test cases.

Concerning the single best evaluation method its performance is at the same level as the n-best re-ranking approach using RFTag and lemma information. Note, however, that for generating the n-best list a beam search of 100 was used. All other systems shown in Table 5.19 deploy a beam size of 12 for beam search decoding. We assume each of their respective results to slightly improve when decoded with a beam size of 100 as well. However, this is not advisable as the decoding duration increases significantly.

Across multiple experiments, the serialization approach showed promising results for the target as well as the source language factorization and the aforementioned combination of both.

In contrast, the joint decoding of target factors unfortunately did not show any benefits. Attempts to explain the disparate behaviour include the following aspects. Firstly, for joint decoding the attention mechanism is shared among all factors. For the serialization approach, there is conceptually only a single attention mechanism as well, but because each factor token is decoded by its "own" decoding step, possibly factor dependent attention mappings may be learned. Secondly, for the joint decoding approach the dimensionality of the embedding space must be divided among factors, leading to embedding vectors of

Table 5.19.: BLEU scores for best results for main translation task (prediction of target words) of each approach

Dev Set		Test Set								Average Test Set	
2013		2010		2011		2012		2014			
single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
baseline (run 2)											
28.59	28.96	27.11	27.72	28.74	30.43	25.79	27.05	24.36	24.93	26.50	27.53
exp1: n-best list re-ranking: RFTags+Lemmas											
		28.05	28.34	**30.27**	30.87	26.95	27.48	25.15	25.43	**27.61**	28.03
exp2: target serialization: LEM (red) » BPE (sys-id 2.9)											
28.41	29.55	26.62	28.21	29.09	31.19	26.03	27.33	24.78	26.21	26.63	28.24
exp3: joint decoding: BPE | RFTags (sys-id 3.1)											
28.14	28.73	26.84	27.35	28.46	29.77	25.91	26.33	24.42	24.66	26.41	27.03
exp4.a: joint encoding (source factors): BPE | POS (sys-id 4.1)											
28.59	29.09	27.08	27.72	29.13	30.56	25.88	26.87	24.97	25.34	26.77	27.62
exp4.b: source serialization (source factors): POS » LEM » IOB » BPE (sys-id 4.12)											
29.59	29.78	27.01	28.10	28.96	30.65	26.62	27.08	24.68	25.26	26.82	27.77
exp4.c: source&target serialization: (source and target factors): src: POS » LEM » BPE ||| trg: RFT » BPE (sys-id 5.2)											
29.74	**30.50**	27.63	**28.38**	30.03	**31.30**	**27.01**	**27.51**	**25.72**	**26.65**	27.60	**28.46**

reduced dimensionality. Whereas for the serialization approach, each factor may make use of the entire embedding space. Lastly, we assume the serialization approach to virtually increase the depth of the decoder. This could allow dependencies among factors to be better exploited. Whereas the NMT toolkit used in this work did only provide a single hidden layer for the decoder at the time of experimentation. Possibly a single layer is not sufficient for the system to find common knowledge among factors.

In the following we discuss further aspects related to our experimentation setting.

Target language and translation task German is a morphologically rich language and therefore is designated to produce data sparseness. As this was one of our main motivations to integrate additional word level knowledge, we assume the translation direction from English to German to be well suited to highlight possible improvements.

The domain of spoken language the TED talks cover however is possibly more challenging when trying to prove the usefulness of mostly linguistic motivated annotations than working with written language. The RFTagger for example was trained on written news articles. It is questionable whether its claimed performance is fully transferable to spoken language. Also, the talks contain rather vivid language which is less grammatical and therefore possible shows lower performance improvement potential regarding linguistic knowledge.

BPE segmentation Prior to experimentation, we had to decide on how many BPE merge rules to use when building our vocabularies of sub-word units. We thereby followed approaches presented in literature to use approximately 40k vocabulary entries to meet the memory requirements of modern GPUs while still being able to model the most common words at word level (i.e., with a single BPE segment). We now wonder whether a reduced number of merge rules leading to shorter sub-word units that might roughly correlate with morphemes could be beneficial. For example due to the introduction of linguistic information the urge could arise to generate a morphological variant of a target word that simply is not found in the word vocabulary and therefore cannot be generated. Splitting the words to morphemes could help in this situation. But the BPE segmentation approach is not linguistically motivated. Therefore, it is questionable whether it is able to find segments that are useful from a linguistic point of view.

From word factorization to multi-tasking We motivated that the use of dedicated tools to derive the additional knowledge could be beneficial due to levels of dedicated expertise, those tools' design might be influenced by. But on the other hand, these tools also introduce errors due to the tasks they have to solve being intricate likewise. By relying on those tools to alleviate the translation problem, the problem of complexity is to some extent outsourced to be solved by the tools. For example, we stated the translation task to suffer from data sparseness and thereby motivated the usage of lemmas and morphological information. On the other hand, tools that derive the latter face data sparseness issues themselves. Therefore, to some extent the problem is only shifted.

However, from a deep learning standpoint, integrating the additional knowledge acquisition task into the main translation systems architecture could lead to superior results.

Using deep learning techniques has been already shown to be successfully applicable to sequence learning tasks like POS tagging as such (e.g., Graves et al., 2012; Popov, 2016). The next step is to incorporate all models into a single one that learns from the different data sources dedicated to the different types of information. This will also relieve us from the burden to model the additional information at word level. This is dubbed as multi-tasking and when performed at a basic level, can be already conceptually covered with similar solutions to those investigated in this thesis. Thereby, a further problem to incorporate the different data sources into the joint training procedure arises. As the latter exceeds the scope of this thesis we decided to use the word level approach. Also, as to the best of our knowledge, at the time of writing no successful multi-tasking approach exists that covers tagging or parsing tasks of linguistic features for the *target* language.

6. Conclusion

To conclude, we give a summary of this thesis and highlight future work that could not be covered.

6.1. Summary

This work investigated the research question whether the integration of additional knowledge in form of target factors into an attentional encoder-decoder Neural Machine Translation system is able to improve translation performance or whether this information is redundant in this context.

In this work the translation direction English to German was explored, thus target factors were applied for German as target language. Utilized factors were part of speech tags enriched with morphosyntactic information (RFTags), lemmas, Dependency Labels and IOB tags together with words modeled at subword level using BPE segmentation.

We first proposed to investigate dedicated word to factor translation systems to be applied in an n-best list re-ranking study. We then proposed two concepts to integrate target factors, namely a serialization and a joint decoding approach. Moreover we investigated a combination with source factorization. We thereby proposed to use the serialization approach on source side as well.

Evaluation was performed on a low-resource setting. The n-best list re-ranking study showed improvements of about +0.6 Bleu compared to baseline when using the combination of scores produced by systems translating to RFTags and Lemmas respectively. Regarding the serialization of Target Factors best results of +0.7 Bleu above baseline could be achieved when subword units are preceded by lemmas based on an optimized vocabulary.

For the concept of joint decoding, we must state, that no significant improvements over baseline were achievable. We therefore suggest that the decoder is not able to extract the additional information, if the factors are presented in parallel. This finding is consistent with recent literature.

In order to establish basis for comparison we performed a brief evaluation of source factorization. We found our suggestion to transfer the serialization approach to the source language to perform slightly better than the proposed approach found in recent literature. We then combined source and target factorization and found that serialization used on both sides simultaneously lead to our best result of +0.9 Bleu above baseline.

6.2. Future Work

We are interested whether the linguistic information can be more beneficial when using a different word segmentation approach that better correlates with linguistic concepts like morphemes. The system then possibly could generate morphological variants that were not seen during training. Maybe a segmentation that is optimal in that respect could be learned automatically.

Further, as the joint decoding approach could not achieve the same results as the serialization approach, we would like to investigate whether the former would benefit form a deeper decoder architecture comprised of multiple stacked layers.

Instead of utilizing external tools to derive the additional linguistic information for the bilingual training corpus we are interested whether the data those tools were trained on could be integrated into the training procedure of the NMT system directly.

Finally, we believe that in the future the integration of additional knowledge will be vital for neural machine translation systems to gain deep semantic understanding. However this will presumably not be rather simple linguistic information expressed at word level. Instead the future might reveal architectures that are able to learn from highly diverging tasks in order to gain the world knowledge necessary whenever the translation task is not easily solved by low level pattern matching. First approaches are taken by e.g., Kaiser et al. (2017), who combined tasks as heterogeneous as image captioning, speech recognition, machine translation and parsing. These efforts are promising but at the time of writing they unfortunately could not yet lead to state-of-the-art results.

References

Abele, Timo (2014).
"Source Sentence Reordering for English to Japanese Machine Translation". Bachelor Thesis, Karlsruhe Institute of Technology.

Alexandrescu, Andrei and Katrin Kirchhoff (2006).
"Factored neural language models". In: *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. Association for Computational Linguistics, pp. 1–4.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014).
"Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473*.

Bangalore, Srinivas and Aravind K Joshi (1999).
"Supertagging: An approach to almost parsing". In: *Computational linguistics* 25.2, pp. 237–265.

Bentivogli, Luisa, Arianna Bisazza, Mauro Cettolo, and Marcello Federico (2016).
"Neural versus phrase-based machine translation quality: a case study". In: *arXiv preprint arXiv:1608.04631*.

Birch, Alexandra, Miles Osborne, and Philipp Koehn (2007).
"CCG supertags in factored statistical machine translation". In: *Proceedings of the Second Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pp. 9–16.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith (2002).
"The TIGER treebank". In: *Proceedings of the workshop on treebanks and linguistic theories*. Vol. 168.

Brants, Thorsten (2000).
"TnT a statistical part-ofspeech tagger. In Proceedings of the Sixth Applied Natural Language Processing (ANLP-2000)". In: *Seattle, WA*.

Britz, Denny, Anna Goldie, Thang Luong, and Quoc Le (2017).
"Massive exploration of neural machine translation architectures". In: *arXiv preprint arXiv:1703.03906*.

Brown, Peter F, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai (1992).
"Class-based n-gram models of natural language". In: *Computational linguistics* 18.4, pp. 467–479.

Brown, Peter F, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer (1993). "The mathematics of statistical machine translation: Parameter estimation". In: *Computational linguistics* 19.2, pp. 263–311.

Cettolo, Mauro, Christian Girardi, and Marcello Federico (2012). "Wit3: Web inventory of transcribed and translated talks". In: *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*. Vol. 261, p. 268.

Cettolo, Mauro, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico (2015). "The IWSLT 2015 evaluation campaign". In: *Proceedings of the International Workshop on Spoken Language Translation, Da Nang, Vietnam*.

Charniak, Eugene, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz (1993). "Equations for part-of-speech tagging". In: *AAAI*. Vol. 93, pp. 784–789.

Cho, Kyunghyun (2015). "Natural Language Understanding with Distributed Representation". In: *CoRR* abs/1511.07916. URL: http://arxiv.org/abs/1511.07916.

Cho, Kyunghyun, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (2014a). "On the properties of neural machine translation: Encoder-decoder approaches". In: *arXiv preprint arXiv:1409.1259*.

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014b). "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078*.

Clematide, Simon (2013). "A case study in tagging case in German: an assessment of statistical approaches". In: *International Workshop on Systems and Frameworks for Computational Morphology*. Springer, pp. 22–34.

Collins English Dictionary (2017). *Collins English Dictionary - Complete & Unabridged 10th Edition*. URL: http://www.dictionary.com/browse/part-of-speech.

Crego, Josep, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, et al. (2016). "SYSTRAN's Pure Neural Machine Translation Systems". In: *arXiv preprint arXiv:1610.05540*.

Crysmann, Berthold, Silvia Hansen-Schirra, George Smith, and Dorothea Ziegler-Eisele (2005). *TIGER Morphologie-Annotationsschema*. Tech. rep. Technical report, Universität Potsdam, Universität Saarbrücken.

Dorr, Bonnie, Matt Snover, and Nitin Madnani. *Part 5: Machine Translation Evaluation*.

Du Preez, Johan Adam and DM Weber (1998).

"Efficient high-order hidden Markov modelling". PhD thesis. Universiteit van Stellenbosch.

Garcia-Martinez, Mercedes, Loïc Barrault, and Fethi Bougares (2016a).
"Factored Neural Machine Translation". In: *arXiv preprint arXiv:1609.04621*.

Garcia-Martinez, Mercedes, Loïc Barrault, and Fethi Bougares (2016b).
"Factored Neural Machine Translation Architectures". In: *International Workshop on Spoken Language Translation (IWSLT'16)*.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016).
Deep Learning. `http://www.deeplearningbook.org`. MIT Press.

Graves, Alex et al. (2012).
Supervised sequence labelling with recurrent neural networks. Vol. 385. Springer.

Han, Aaron Li-Feng and Derek Fai Wong (2016).
"Machine Translation Evaluation: A Survey". In: *arXiv preprint arXiv:1605.04515*.

Hoang, Cong Duy Vu, Reza Haffari, and Trevor Cohn (2016).
"Improving Neural Translation Models with Linguistic Factors". In: *Proceedings of the Australasian Language Technology Association Workshop 2016*, pp. 7–14.

Hochreiter, Sepp and Jürgen Schmidhuber (1997).
"Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Huang, Xuedong, Alex Acero, Hsiao-Wuen Hon, and Raj Foreword By-Reddy (2001).
Spoken language processing: A guide to theory, algorithm, and system development. Prentice hall PTR.

Junczys-Dowmunt, Marcin, Tomasz Dwojak, and Hieu Hoang (2016).
"Is neural machine translation ready for deployment? a case study on 30 translation directions". In: *arXiv preprint arXiv:1610.01108*.

Jurafsky, Dan and James H. Martin (2009).
Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition. 2. ed., Pearson International Edition. Prentice Hall series in artificial intelligence. Upper Saddle River, NJ [u.a.]: Prentice Hall, Pearson Education International. ISBN: 0-13-504196-1; 978-0-13-504196-3.

Jurafsky, Dan and James H Martin (2014).
Speech and language processing. Vol. 3. Pearson London.

Kaiser, Lukasz, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit (2017).
"One Model To Learn Them All". In: *arXiv preprint arXiv:1706.05137*.

Kalchbrenner, Nal and Phil Blunsom (2013).
"Recurrent Continuous Translation Models." In: *EMNLP*. Vol. 3. 39, p. 413.

Koehn, Philipp, Marcello Federico, Wade Shen, Nicola Bertoldi, Ondrej Bojar, Chris Callison-Burch, Brooke Cowan, Chris Dyer, Hieu Hoang, Richard Zens, et al. (2006).

"Open source toolkit for statistical machine translation: Factored translation models and confusion network decoding". In: *Final Report of the 2006 JHU Summer Workshop*.

Koehn, Philipp and Hieu Hoang (2007).
"Factored Translation Models." In: *EMNLP-CoNLL*, pp. 868–876.

Koehn, Philipp (2009).
Statistical machine translation. Cambridge University Press.

Lebret, Rémi Philippe (2016).
"Word Embeddings for Natural Language Processing". PhD thesis.

Liu, Chang, Daniel Dahlmeier, and Hwee Tou Ng (2011).
"Better evaluation metrics lead to better machine translation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 375–384.

Luong, Minh-Thang, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser (2015).
"Multi-task sequence to sequence learning". In: *arXiv preprint arXiv:1511.06114*.

Maier, Wolfgang, Sandra Kübler, Daniel Dakota, and Daniel Whyatt (2014).
"Parsing German: How Much Morphology Do We Need?" In: *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pp. 1–14.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013a).
"Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*.

Mikolov, Tomas, Martin Karafiát, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur (2010).
"Recurrent neural network based language model." In: *Interspeech*. Vol. 2, p. 3.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013b).
"Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013c).
"Linguistic regularities in continuous space word representations." In: *hlt-Naacl*. Vol. 13, pp. 746–751.

Nadejde, Maria, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch (2017).
"Syntax-aware Neural Machine Translation Using CCG". In: *arXiv preprint arXiv:1702.01147*.

Neunerdt, Melanie (2016).
"Part-of-speech tagging and detection of social media texts". PhD thesis. Dissertation, RWTH Aachen, 2015.

Niehues, Jan and Eunah Cho (2017).

"Exploiting Linguistic Resources for Neural Machine Translation Using Multi-task Learning". In: *arXiv preprint arXiv:1708.00993*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002).
"BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pp. 311–318.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013).
"On the difficulty of training recurrent neural networks". In: *International Conference on Machine Learning*, pp. 1310–1318.

Popov, Alexander (2016).
"Deep Learning Architecture for Part-of-Speech Tagging with Word and Suffix Embeddings". In: *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*. Springer, pp. 68–77.

Pouget-Abadie, Jean, Dzmitry Bahdanau, Bart van Merrienboer, Kyunghyun Cho, and Yoshua Bengio (2014).
"Overcoming the curse of sentence length for neural machine translation using automatic segmentation". In: *arXiv preprint arXiv:1409.1257*.

Powers, David MW (2007).
Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation, School of Informatics and Engineering, Flinders University, Adelaide, Australia. Tech. rep. TR SIE-07-001, Journal of Machine Learning Technologies 2: 1 37-63. https://dl-web. dropbox. com/get/Public/201101-Evaluation_JMLT_Postprint-Colour. pdf.

Radford, Andrew (2006).
Minimalist syntax revisited. Citeseer.

Al-Rfou, Rami, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frederic Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, et al. (2016).
"Theano: A Python framework for fast computation of mathematical expressions". In: *arXiv preprint*.

Sajjad, Hassan and Helmut Schmid (2009).
"Tagging Urdu text with parts of speech: A tagger comparison". In: *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 692–700.

Schiller, Anne, Simone Teufel, and Christine Thielen (1995).
"Guidelines f ur das Tagging deutscher Textcorpora mit STTS". In: *Universitäten Stuttgart und Tübingen*.

Schmid, Helmut (1994a).
"Part-of-speech tagging with neural networks". In: *Proceedings of the 15th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pp. 172–176.

Schmid, Helmut (1994b).
Probabilistic part-of-speech tagging using decision trees. In proceedings of international conference on new methods in language processing,(pp. 1-9), Access date: 09.11. 2012.

Schmid, Helmut (1995).
"Improvements in part-of-speech tagging with an application to German". In: *In proceedings of the acl sigdat-workshop*. Citeseer.

Schmid, Helmut and Florian Laws (2008).
"Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging". In: *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, pp. 777–784.

Schneider, Gerold, Michael Hess, and Paola Merlo (2008).
"Hybrid long-distance functional dependency parsing". In: *Unpublished doctoral dissertation). University of Zurich: Institute of Computational Linguistics, Zurich, Switzerland.*

Sennrich, Rico and Barry Haddow (2016).
"Linguistic input features improve neural machine translation". In: *arXiv preprint arXiv:1606.02892*.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2015a).
"Improving neural machine translation models with monolingual data". In: *arXiv preprint arXiv:1511.06709*.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2015b).
"Neural machine translation of rare words with subword units". In: *arXiv preprint arXiv:1508.07909*.

Sennrich, Rico, Gerold Schneider, Martin Volk, and Martin Warin (2009).
"A new hybrid dependency parser for German". In: *Proceedings of the German Society for Computational Linguistics and Language Technology* 115, p. 124.

Sennrich, Rico, Martin Volk, and Gerold Schneider (2013).
"Exploiting Synergies Between Open Resources for German Dependency Parsing, POS-tagging, and Morphological Analysis." In: *RANLP*, pp. 601–609.

Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. (2017).
"Nematus: a toolkit for neural machine translation". In: *arXiv preprint arXiv:1703.04357*.

Shi, Xing, Inkit Padhi, and Kevin Knight (2016).
"Does String-Based Neural MT Learn Source Syntax?" In: *EMNLP*, pp. 1526–1534.

Song, Xingyi, Trevor Cohn, and Lucia Specia (2013).
"BLEU deconstructed: Designing a better MT evaluation metric". In: *International Journal of Computational Linguistics and Applications* 4.2, pp. 29–44.

Stahlberg, Felix (2016).

"Guided Neural Machine Translation". PhD thesis. University of Cambridge.

Steedman, Mark (2000).
The syntactic process. Vol. 24. MIT Press.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014).
"Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, pp. 3104–3112.

Telljohann, Heike, Erhard Hinrichs, Sandra Kübler, and Ra Kübler (2004).
"The TüBa-D/Z treebank: Annotating German with a context-free backbone". In: *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004*. Citeseer.

Waibel, Alex, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang (1989).
"Phoneme recognition using time-delay neural networks". In: *IEEE transactions on acoustics, speech, and signal processing* 37.3, pp. 328–339.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. (2016).
"Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144*.

Yang, Nan, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu (2013).
"Word Alignment Modeling with Context Dependent Deep Neural Network." In: *ACL (1)*, pp. 166–175.

Zeiler, Matthew D (2012).
"ADADELTA: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701*.

Zhang, Jiajun and Chengqing Zong (2015).
"Deep Neural Networks in Machine Translation: An Overview". In: *IEEE Intelligent Systems* 5.30, pp. 16–25.

Zhou, Rong and Eric A Hansen (2005).
"Beam-Stack Search: Integrating Backtracking with Beam Search." In: *ICAPS*, pp. 90–98.

A. Appendix

A.1. N-Best List Re-ranking

Table A.1.: Bleu scores for Re-Ranking

Re-Ranking	Test Set								Average Test Set	
	2010		2011		2012		2014			
	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
baseline (run 1. beam 12)	26.71	27.90	29.53	30.11	25.65	26.74	24.19	24.78	26.52	27.38
baseline (run 1. beam 100)	26.91	27.93	29.53	30.18	25.76	26.85	24.21	24.76	26.88	27.43
RFTags	27.16	28.26	29.74	30.34	26.25	27.22	25.05	25.27	27.05	27.77
RFTags. dims reduced	27.39	28.12	29.80	30.63	26.28	26.96	24.63	25.26	27.03	27.74
Lemmas	27.90	28.15	30.25	30.55	26.54	27.05	24.78	25.14	27.37	27.72
RFTags+Lemmas	28.05	28.34	30.27	30.87	26.95	**27.48**	25.15	**25.43**	27.61	**28.03**
baseline (run 2)	27.84	28.16	30.53	30.75	26.38	26.97	24.94	25.04	27.42	27.73
baseline (run 3)	27.81	28.42	30.49	30.79	26.89	27.08	25.01	25.13	27.55	27.86
baseline (run2 + run3)	**28.20**	**28.45**	**30.76**	**30.98**	**27.06**	27.10	**25.22**	25.39	**27.81**	27.98
Average Random	24.15	25.03	26.56	27.37	22.78	23.63	21.89	22.41	23.84	24.61
Variance	0.02	0.09	0.04	0.02	0.02	0.05	0.03	0.07	0.03	0.06
WORST	16.70	18.86	17.44	19.60	17.58	19.09	17.21	19.30	17.23	19.21

A.2. Serialization of Target Factors

Table A.2.: BLEU scores for main translation task (prediction of target words) – approach Serializing

	Serializing		Development Set		Test Set								Average Test Set	
	WORDS		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	baseline (run2)		28.59	28.96	27.11	27.72	28.74	30.43	25.79	27.05	24.36	24.93	26.50	27.53
2.1	RFTags » BPE	single	29.04	30.04	27.38	27.86	29.04	30.49	26.09	26.83	24.78	25.53	26.82	27.68
2.2	RFTags » BPE	copy	28.78	29.57	26.11	27.21	28.78	29.78	25.53	26.51	24.32	25.39	26.19	27.22
2.3	BPE » RFTags	single	28.80	29.29	26.74	28.16	28.65	30.46	**26.16**	26.80	24.36	25.49	26.48	27.73
2.4	BPE » RFTags	copy	28.55	29.14	26.88	28.21	28.82	30.87	26.02	26.72	24.48	25.03	26.55	27.71
2.5	LEM » BPE	single	29.17	29.93	26.84	28.13	28.15	30.16	25.75	27.17	24.62	25.58	26.34	27.76
2.6	LEM » BPE	copy	28.14	29.32	25.63	27.60	29.04	29.99	25.19	26.95	23.55	24.84	25.85	27.35
2.7	BPE » LEM	copy	28.54	29.48	26.68	27.95	28.22	30.21	25.76	27.14	23.99	25.19	26.16	27.62
2.8	LEM (red) » BPE	single	28.50	29.48	26.65	28.04	28.88	30.46	25.96	27.05	24.41	25.22	26.48	27.69
2.9	LEM (red) » BPE	copy	28.41	29.55	26.62	28.21	29.09	**31.19**	26.03	**27.33**	24.78	**26.21**	26.63	**28.24**
2.10	DEP » BPE	single	**29.32**	**30.06**	26.71	27.98	29.28	30.58	25.91	27.16	**24.83**	25.56	26.68	27.82
2.11	IOB » BPE	single	28.81	29.63	27.03	28.27	29.04	30.88	26.10	27.25	24.53	25.26	26.68	27.92
2.12	DUM » BPE	single	28.69	29.12	27.33	**28.77**	29.81	30.94	25.81	26.76	24.43	25.29	26.85	27.94
2.13	DUM » BPE	copy	29.06	29.60	**27.61**	28.47	**30.01**	30.96	26.03	26.55	24.60	24.93	**27.06**	27.73

Table A.3.: BLEU scores for prediction of RFTags – approach Serializing

	Serializing		Development Set		Test Set								Average Test Set	
	RFTags		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	RFTags (baseline)		32.60	33.80	30.48	**32.34**	31.77	33.94	29.61	31.08	28.48	29.20	30.09	31.64
2.1	RFTags » BPE	single	32.97	**33.97**	**31.64**	32.31	33.06	34.45	**30.12**	31.01	**28.54**	29.28	**30.84**	**31.76**
2.2	RFTags » BPE	copy	31.10	32.66	29.52	30.61	31.70	32.73	28.72	30.17	27.64	28.52	29.40	30.51
2.3	BPE » RFTags	single	32.60	33.04	30.89	32.31	32.75	34.25	29.69	30.50	27.71	29.07	30.26	31.53
2.4	BPE » RFTags	copy	31.62	32.50	29.71	31.47	31.30	33.34	28.70	30.31	27.12	28.15	29.21	30.82

Table A.4.: Bleu scores for prediction of Lemmas – approach Serializing

	Serializing		Development Set		Test Set								Average Test Set	
	Lemmas		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	Lemmas (baseline)		32.37	33.23	**30.50**	31.13	**32.80**	**33.83**	**30.11**	**30.84**	**28.57**	**29.13**	**30.50**	**31.23**
2.5	Lemmas » BPE	single	**32.89**	**33.45**	30.02	**31.29**	31.89	33.72	29.28	30.50	28.15	29.02	29.84	31.13
2.6	Lemmas » BPE	copy	31.41	32.56	28.98	30.49	32.03	32.80	28.45	30.03	27.28	28.34	29.19	30.42
2.7	BPE » LEM	copy	31.57	32.53	29.60	30.82	31.36	33.06	28.79	30.01	27.38	28.75	29.28	30.66

Table A.5.: BLEU scores for prediction of Dependency Labels – approach Serializing

	Serializing		Development Set		Test Set								Average Test Set	
	DEP		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	DEP (baseline)		46.35	46.57	44.88	45.74	44.86	45.75	42.81	43.27	43.28	43.43	43.96	44.55
2.10	DEP » BPE	single	46.99	47.43	45.20	45.87	46.51	47.57	43.32	43.87	43.37	43.52	44.60	45.21

Table A.6.: Bleu scores for prediction of IOB tags – approach Serializing

Serializing			Development Set		Test Set								Average Test Set	
	IOB		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	IOB (baseline)		69.64	56.54	76.83	60.58	77.14	66.69	76.00	65.44	72.67	58.50	75.66	62.80
2.11	IOB » BPE	single	84.26	84.16	83.52	83.37	83.53	83.51	82.49	82.58	82.27	81.89	82.95	82.84

A.3. Joint Decoding of Target Factors

Table A.7.: Bleu scores for main translation task (prediction of target words) – approach Joint Decoding of Target Factors

	Joint Decoding		Development Set		Test Set								Average Test Set	
	WORDS		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	baseline (run 2)		**28.59**	**28.96**	**27.11**	**27.72**	28.74	**30.43**	25.79	**27.05**	24.36	24.93	26.50	**27.53**
3.1	BPE | RFT	copy	28.14	28.73	26.84	27.35	28.46	29.77	25.91	26.33	24.42	24.66	26.41	27.03
3.2	BPE | LEM	copy	28.00	28.58	25.90	27.02	28.15	29.56	25.08	25.98	23.47	24.32	25.65	26.72
3.3	BPE | DEP	copy	28.22	28.57	26.71	27.26	28.04	29.35	25.32	25.83	24.06	24.39	26.03	26.71
3.4	BPE | IOB	IOB	28.34	28.73	26.59	27.11	28.92	29.75	25.47	26.15	24.04	24.58	26.26	26.90
3.5	BPE | RFT_IOB	IOB	28.47	28.73	26.16	27.06	28.98	30.17	25.23	25.88	23.58	24.46	25.99	26.89
3.6	BPE | RFT | IOB | LEM	copy	26.73	27.04	25.52	26.16	27.65	28.12	24.33	24.66	22.83	23.14	25.08	25.52
3.7	BPE | RFT | IOB | LEM (Beam100)	copy	26.83		25.83		27.73		24.48		32.11		25.29	
3.8	BPE | NUM	-	28.13	28.48	26.76	27.54	**29.13**	29.70	26.03	26.30	24.22	**25.06**	**26.54**	27.15
3.9	BPE | RFT (red. dec. dim. 1)	copy	28.37	28.49	26.65	27.26	28.84	29.67	25.59	26.14	23.97	24.68	26.26	26.94
3.10	BPE | RFT (red. dec. dim. 2)	copy	27.89	28.79	26.18	27.11	28.72	29.50	25.37	26.13	23.77	24.20	26.01	26.74
3.11	BPE | RFT (red. dec. dim. 3)	copy	28.08	28.65	26.43	27.07	28.68	29.51	**26.16**	26.59	24.13	24.25	26.35	26.86
3.12	BPE | RFT | IOB | LEM (red. emb. dim.)	copy	26.91	27.37	25.18	25.84	26.93	27.90	24.04	24.33	22.54	23.26	24.67	25.33
3.13	BPE (emb 10) | RFT | IOB | LEM	copy	26.02	26.72	24.26	25.42	25.99	27.03	23.68	24.44	22.50	22.78	24.11	24.92
3.14	BPE delayed +1 | RFT | IOB | LEM	copy	24.52	25.11	22.49	23.21	24.05	25.36	21.29	22.06	19.74	20.68	21.89	22.83

Table A.8.: BLEU scores for prediction of RFTags – approach Joint Decoding of Target Factors

	Joint Decoding		Development Set		Test Set								Average Test Set	
	RFTags		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	RFT (baseline)		**32.60**	**33.80**	**30.48**	**32.34**	**31.77**	**33.94**	**29.61**	**31.08**	**28.48**	**29.20**	**30.09**	**31.64**
3.1	BPE | RFT	copy	31.11	32.21	29.63	30.46	30.96	32.23	28.81	29.13	27.82	28.05	29.31	29.97
3.5	BPE | RFT_IOB	IOB	29.55	30.40	28.09	28.78	29.71	31.02	27.03	27.80	25.94	26.90	27.69	28.63
3.6	BPE | RFT | IOB | LEM	copy	30.46	31.00	29.27	29.97	30.72	31.05	28.39	28.97	26.69	27.53	28.77	29.38
3.9	BPE | RFT (red. dec. dim. 1)	copy	31.27	31.83	29.47	30.24	31.64	32.16	28.68	29.21	27.16	28.21	29.24	29.96
3.10	BPE | RFT (red. dec. dim.2)	copy	30.51	31.58	29.59	30.58	30.82	31.95	28.14	28.96	26.97	27.98	28.88	29.87
3.11	BPE | RFT (red. dec. dim. 3)	copy	30.43	31.77	28.90	30.05	31.07	31.90	28.63	29.37	26.69	27.51	28.82	29.71
3.12	BPE | RFT | IOB | LEM (red. emb. dim.)	copy	31.04	32.19	29.22	30.24	30.81	31.70	28.07	28.91	27.12	27.89	28.81	29.69
3.13	BPE (emb 10) | RFT | IOB | LEM	copy	29.55	30.19	27.78	28.80	29.07	29.99	27.12	27.80	25.64	26.38	27.40	28.24
3.14	BPE delayed +1 | RFT | IOB | LEM	copy	26.92	27.39	25.97	26.34	26.86	28.06	24.46	25.56	22.26	23.39	24.89	25.84

Table A.9.: Bleu scores for prediction of Lemmas – approach Joint Decoding of Target Factors

	Joint Decoding		Development Set		Test Set								Average Test Set	
	Lemmas		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	LEM (baseline)		**32.37**	**33.23**	**30.50**	**31.13**	**32.80**	**33.83**	**30.11**	**30.84**	**28.57**	**29.13**	**30.50**	**31.23**
3.2	BPE | LEM	copy	31.20	32.26	28.96	30.26	31.60	32.86	28.65	29.78	27.82	28.41	29.26	30.33
3.6	BPE | RFT | IOB | LEM	copy	30.63	30.82	28.85	29.61	30.87	31.55	28.16	28.75	27.10	27.33	28.75	29.31
3.12	BPE | RFT | IOB | LEM (red. emb. dim.)	copy	31.41	32.17	29.25	30.27	31.73	32.52	28.55	29.06	26.72	27.59	29.06	29.86
3.13	BPE (emb 10) | RFT | IOB | LEM	copy	29.24	30.00	27.56	28.42	29.32	30.18	27.12	28.02	25.20	26.20	27.30	28.21
3.14	BPE delayed +1 | RFT | IOB | LEM	copy	26.94	27.72	25.38	26.10	26.64	28.17	24.43	25.35	22.01	23.17	24.62	25.70

Table A.10.: BLEU scores for prediction of Dependency Labels – approach Joint Decoding of Target Factors

	Joint Decoding		Development Set		Test Set								Average Test Set	
	DEP		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	Dependency Lables (baseline)		**46.35**	**46.57**	**44.88**	**45.74**	**44.86**	**45.75**	**42.81**	**43.27**	**43.28**	**43.43**	**43.96**	**44.55**
3.3	BPE | DEP	copy	44.54	45.32	42.78	43.73	43.77	44.37	40.93	41.95	41.31	41.70	42.20	42.94

Table A.11.: BLEU scores for prediction of IOB tags – approach Joint Decoding of Target Factors

	Joint Decoding		Development Set		Test Set								Average Test Set	
	IOB		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	IOB (baseline)		69.64	56.54	76.83	60.58	77.14	66.69	76.00	65.44	72.67	58.50	75.66	62.80
3.4	BPE | IOB	IOB	**83.41**	**83.51**	**82.84**	**83.06**	**83.49**	**83.54**	**82.13**	**82.32**	**81.44**	**81.92**	**82.48**	**82.71**
3.13	BPE (emb 10) | RFT | IOB | LEM	IOB	82.03	82.23	81.06	81.50	81.14	81.65	80.09	80.55	78.76	79.60	80.26	80.83

A.4. Combination With Source Factorization

A.4.1. Source Factorization

Table A.12.: BLEU scores for main translation task (prediction of target words) – approach Source Factorization, Joint Encoding

	Source Factored		Development Set		Test Set								Average Test Set	
	WORDS		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	baseline (run 2)		28.59	28.96	**27.11**	27.72	28.74	30.43	25.79	**27.05**	24.36	24.93	26.50	27.53
4.1	BPE | POS	copy	28.59	29.09	27.08	27.72	**29.13**	**30.56**	25.88	26.87	**24.97**	25.34	**26.77**	**27.62**
4.2	BPE | LEM	copy	28.82	28.82	26.94	**28.05**	28.21	29.65	**26.03**	26.78	24.71	25.19	26.47	27.42
4.3	BPE | IOB	IOB	28.28	28.92	27.03	27.78	28.77	29.71	25.98	26.71	24.14	25.20	26.48	27.35
4.4	BPE | POS | LEM	copy	28.40	29.16	26.43	27.85	28.93	30.47	25.96	26.61	24.31	25.23	26.41	27.54
4.5	BPE | POS | LEM | IOB	copy	**28.93**	**29.64**	26.45	27.57	28.80	29.95	25.70	26.98	24.69	**25.45**	26.41	27.49

Table A.13.: BLEU scores for main translation task (prediction of target words) – approach Source Factorization, Serialization of Source Factors

	Alternating		Development Set		Test Set								Average Test Set	
	WORDS		2013		2010		2011		2012		2014			
sys-id		SWM	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	baseline (run 2)		28.59	28.96	27.11	27.72	28.74	30.43	25.79	27.05	24.36	24.93	26.50	27.53
4.6	POS » BPE	single	29.07	29.04	26.87	27.85	**29.90**	**30.78**	26.42	26.98	24.72	25.26	**26.98**	27.72
4.7	POS » BPE	copy	28.62	29.51	26.73	27.43	28.57	30.13	25.91	26.90	24.67	**25.59**	26.47	27.51
4.8	LEM » BPE	single	28.92	29.08	26.37	27.98	28.32	29.57	25.37	26.59	24.75	25.31	26.20	27.36
4.9	LEM » BPE	copy	29.01	29.50	26.86	**28.26**	28.72	30.54	26.34	26.90	24.56	25.33	26.62	27.76
4.10	IOB » BPE	-	28.47	29.41	26.95	27.82	28.59	30.39	25.93	26.65	24.24	25.23	26.43	27.52
4.11	POS » LEM » BPE	single	29.48	29.52	26.60	27.82	28.37	30.58	26.24	26.90	**24.87**	25.52	26.52	27.71
4.12	POS » LEM » IOB » BPE	single	**29.59**	**29.78**	27.01	28.10	28.96	30.65	**26.62**	**27.08**	24.68	25.26	26.82	**27.77**
4.13	DUM » BPE	copy	28.63	29.01	**27.12**	27.55	29.20	30.61	26.20	26.91	24.41	25.08	26.73	27.54

A.4.2. Combination of Source and Target Factor Approaches

Table A.14.: Bleu scores for main translation task (prediction of target words), approach Combination with Source Factorization

	Serialization of Source and Target Side				Development Set		Test Set								Average Test Set	
	WORDS				2013		2010		2011		2012		2014			
sys-id	Source		Target		single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	baseline (run 2)				28.59	28.96	27.11	27.72	28.74	30.43	25.79	27.05	24.36	24.93	26.50	27.53
5.1	BPE | POS | LEM	copy	RFT » BPE	single	29.45	30.27	27.08	28.23	29.13	30.60	26.18	27.43	25.59	26.27	27.00	28.13
5.2	POS » LEM » BPE	single	RFT » BPE	single	**29.74**	**30.50**	**27.63**	**28.38**	**30.03**	**31.30**	**27.01**	27.51	**25.72**	**26.65**	**27.60**	**28.46**
5.3	BPE | POS | LEM	copy	BPE | RFT	copy	28.31	29.15	26.57	27.37	28.51	30.13	26.22	27.16	24.62	25.22	26.48	27.47
5.4	POS » LEM » BPE	single	BPE | RFT	copy	28.89	29.22	26.98	27.75	29.32	30.65	25.87	27.26	24.8	25.36	26.74	27.76
5.5	BPE | POS | LEM	copy	LEM » BPE	single	29.19	30.36	26.5	28.25	28.8	30.77	26.3	**27.86**	24.86	25.86	26.62	28.19

Table A.15.: BLEU scores for prediction of (target) RFTags, approach Combination with Source Factorization

	Serialization of Source and Target Side				Development Set		Test Set								Average Test Set	
	RFTags				2013		2010		2011		2012		2014			
sys-id	Source		Target		single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	RFTags (Baseline)				32.60	33.80	30.48	32.34	31.77	33.94	29.61	31.08	28.48	29.20	30.09	31.64
5.1	BPE | POS | LEM	copy	RFT » BPE	single	**33.38**	**34.06**	31.77	32.82	33.91	34.92	30.59	31.75	**29.76**	30.40	31.51	32.47
5.2	POS » LEM » BPE	single	RFT » BPE	single	33.13	33.93	**32.16**	**32.91**	**34.00**	**35.65**	**31.21**	31.68	29.53	30.80	**31.73**	**32.76**
5.3	BPE | POS | LEM	copy	BPE | RFT	copy	32.11	32.92	29.78	30.68	31.29	32.46	28.91	30.03	27.96	29.04	29.49	30.55
5.4	POS » LEM » BPE	single	BPE | RFT	copy	31.91	32.89	29.93	31.32	31.29	33.27	29.23	30.38	28.37	29.02	29.71	31.00

Table A.16.: Bleu scores for prediction of (target) Lemmas, approach Combination with Source Factorization

	Serialization of Source and Target Side			Development Set		Test Set								Average Test Set	
	LEM			2013		2010		2011		2012		2014			
sys-id	Source		Target	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble	single best	ensemble
	LEM (Baseline)			32.37	33.23	**30.50**	31.13	**32.80**	33.83	**30.11**	30.84	**28.57**	**29.13**	**30.50**	31.23
5.5	BPE | POS | LEM	copy	LEM » BPE	single	**32.67**	**33.56**	29.80	**31.26**	32.48	**34.47**	29.72	**31.01**	28.31	29.02	30.08	**31.44**