What Makes Word-level Neural Machine Translation Hard: A Case Study on English-German Translation

Fabian Hirschmann **Jinseok Nam** **Johannes Fürnkranz**
Knowledge Engineering Group
Technische Universität Darmstadt
Darmstadt, Germany

Abstract

Traditional machine translation systems often require heavy feature engineering and the combination of multiple techniques for solving different subproblems. In recent years, several end-to-end learning architectures based on recurrent neural networks have been proposed. Unlike traditional systems, Neural Machine Translation (NMT) systems learn the parameters of the model and require only minimal preprocessing. Memory and time constraints allow to take only a fixed number of words into account, which leads to the out-of-vocabulary (OOV) problem. In this work, we analyze why the OOV problem arises and why it is considered a serious problem in German. We study the effectiveness of compound word splitters for alleviating the OOV problem, resulting in a 2.5+ BLEU points improvement over a baseline on the WMT'14 German-to-English translation task. For English-to-German translation, we use target-side compound splitting through a special syntax during training that allows the model to merge compound words and gain 0.2 BLEU points.

1 Introduction

In the field of machine translation, traditional methods depend on a careful design of useful features obtained by analyzing linguistic properties of the translation tasks. Inspired by recent success of methods that learn vector representation of words from large corpora in an unsupervised way, Devlin et al. (2014) have shown that word representations learned by neural language models can be effectively used as components in a complex translation system. By contrast, *Neural Machine Translation (NMT)* models are end-to-end learning systems that tune model parameters so that the desired output sentence is generated for a given input sentence of arbitrary length with minimal processing steps such as tokenization. Most of recently proposed NMT models are based on the encoder-decoder framework (Sutskever et al., 2014; Cho et al., 2014), where a source sentence is mapped into a fixed-size vector that preserves both the syntactic and semantic structure of the source sentence, from which a decoder starts with generating a sequence of words in a target language. Bahdanau et al. (2015) extend the vanilla encoder-decoder NMT framework by adding a small feed-forward neural network which learns which word in the source sentence is relevant for predicting the next word in the target sequence. It has been shown that the performance of such *soft-attention* NMTs stays consistent as the sequence length increases.

Although NMT models have been applied successfully to translation tasks, it is still challenging to handle *out-of-vocabulary (OOV)* words because only a small number of words are considered to reduce computational overhead. All words not in the vocabulary are assigned to a single special token, e.g., $\langle$UNK$\rangle$ in our case. In order to address the OOV problem, Jean et al. (2015) further extend the model of Bahdanau et al. (2015) with importance sampling so that it can hold a larger vocabulary without increasing training complexity. Luong et al. (2015) address the OOV problem by looking up unknown words in an automatically generated dictionary, and use an external word aligner to map words in the target sequence to ones in the source sequence.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3199–3208, Osaka, Japan, December 11-17 2016.

Regarding OOV words, a fundamental problem of NMT models that take words as inputs is that having a larger vocabulary cannot be a solution for morphologically rich languages such as German and Czech due to the proliferation of word forms resulting from the addition of affixes to word stems. This means that even though we include millions of words in the vocabulary, quite a significant ratio of words remains unknown tokens. We will discuss such properties of German in Section 3.

Therefore, instead of using words as only inputs and outputs, there have been several approaches which take into account more fine-grained units such as characters. Based on the fact that rare words are often composed of frequent words, Sennrich et al. (2016) build a vocabulary of subunits instead of words and view a sentence as a sequence of subunits. Such a vocabulary typically includes meaningful linguistic units, e.g., prefixes, suffixes and frequent stems, but it may also include less meaningful units because the vocabulary building process solely relies on their frequencies. While learning directly from characters of both source and target sentences is interesting since the OOV problem is not an issue anymore, it is more difficult and time-consuming to train such models (Ling et al., 2015). As a compromise, (Luong and Manning, 2016) propose a hybrid model that uses both words and characters. Similarly, (Chung et al., 2016) suggest the use of a character-level decoder which takes words or subunits as inputs but outputs character sequences.

According to the recent developments in NMT, it is highly likely that fine-grained units enable us to get better neural translation models. In this paper we, hence, explore the problems that are typically encountered by word-level neural translation systems, how we can avoid such problems and what remains problematic. We will start with a brief explanation of the word-level NMT architecture proposed by (Bahdanau et al., 2015) in the next section.

2 Background: Neural Machine Translation

Consider that we have a pair of a source sentence and its translation in the target language, e.g., (*Wie geht es dir?*, *How are you?*) when translating German sentences to English. If a word is treated as an atomic unit of translation, we can define both source and target sentences as a sequence of words such that $\boldsymbol{x} = \{x_j\}_{j=1}^{Tx}$ and $\boldsymbol{y} = \{y_i\}_{i=1}^{Ty}$ where T_x is the number of words in the source sentence and T_y is that of the target words. The goal of learning translation models can be expressed as maximizing the joint probability $p\left(\boldsymbol{y}|\boldsymbol{x}\right)$ of the target words $\boldsymbol{y} = \{y_1, \cdots, y_{T_y}\}$ conditioned on the source words $\boldsymbol{x} = \{x_1, \cdots, x_{T_x}\}$. We can factorize it into a product of conditional probabilities of a single target word:

$$p\left(\boldsymbol{y}|\boldsymbol{x}\right) = \prod_{i=1}^{T_y} p\left(y_i|\boldsymbol{y}_{<i}, \boldsymbol{x}\right) \tag{1}$$

where $\boldsymbol{y}_{<i} = \{y_1, \cdots, y_{i-1}\}$ denotes a set of words preceding a word at position i in the target sentence. Each conditional $p(y_i|\boldsymbol{y}_{<i}, \boldsymbol{x})$ can be defined in a parameterized form by

$$p(y_i = k|\boldsymbol{y}_{<i}, \boldsymbol{x}) = \frac{\exp(s_{ik})}{\sum_{v=1}^{V} \exp(s_{iv})} \tag{2}$$

where s_{ik} is a score for predicting the k-th word in the target vocabulary as a word at position i. Since in most cases T_x and T_y are variable across sentence pairs as well as $T_x \neq T_y$ in a single pair of sentences, we use the *encoder-decoder* architecture, also known as *sequence-to-sequence* learning (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015), where two RNNs are trained jointly to handle such variable inputs and outputs. Using the encoder-decoder RNNs the score s_{ik} can be defined by $s_{ik} = f\left(y_{i-1}, \mathbf{h}_i, \mathbf{c}_i\right)$. One RNN, the so-called *encoder*, computes the hidden state $\overline{\mathbf{h}}_j$ of a source word x_j given previous hidden state as $\overline{\mathbf{h}}_j = \text{RNN}_{enc}(\mathbf{x}_j, \overline{\mathbf{h}}_{j-1})$ where $\mathbf{x}_j$ is vector representation of a word x_j. Similarly, we obtain a hidden state $\mathbf{h}_i$ to be used for predicting y_i using the other, so-called *decoder* RNN as $\mathbf{h}_i = \text{RNN}_{dec}\left(\mathbf{y}_{i-1}, \mathbf{h}_{i-1}, \mathbf{c}_i\right)$ where $\mathbf{c}_i = \sum_{j=1}^{T_x} \alpha_{ij} \overline{\mathbf{h}}_j$ s.t. $\sum_j \alpha_{ij} = 1$ is a context vector, which is a weighted sum of input hidden states. Moreover, we also learn the *attention weights* $\alpha_{ij} = f_{att}\left(\mathbf{h}_{i-1}, \overline{\mathbf{h}}_j\right)$. The decoder's initial hidden state $\mathbf{h}_0$ is initialized by the encoder, e.g., using the last

POS/NE	German		English	
	Coverage	Ratio	Coverage	Ratio
TRUNC	36.84%	0.12%		
I-PER	37.03%	2.21%	41.69%	2.43%
I-ORG	51.67%	1.33%	76.70%	2.14%
I-LOC	63.17%	1.30%	76.18%	1.56%
ITJ/UH	66.67%	0.01%	70.00%	0.01%
NN	68.73%	20.50%	93.25%	14.75%
VVIZU	70.37%	0.17%		
FM/FW	70.76%	0.27%	61.90%	0.03%
ADJ	78.99%	7.43%	90.49%	6.89%
VVFIN	86.08%	4.53%		
VVPP	87.13%	2.29%		
CARD/CD	87.62%	1.65%	92.01%	1.85%
VVINF	90.11%	1.80%		

Table 1: Statistics of different tag sets.

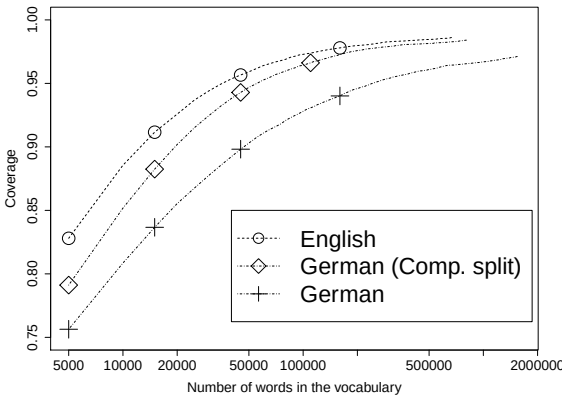

Figure 1: Coverage of the test set w.r.t. the size of vocabulary.

hidden state $\overline{\mathbf{h}}_{T_x}$ of the source sentence $\boldsymbol{x}$ while we can set the initial hidden state of the encoder $\overline{\mathbf{h}}_0 = \mathbf{0}$. For further details, see (Bahdanau et al., 2015).

Given a corpus of N training examples $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$, we can iteratively update the model parameters θ by using the gradient of the negative log-likelihood given by

$$\frac{\partial}{\partial\theta}\mathcal{L}(\theta;(\boldsymbol{x}^{(n)},\boldsymbol{y}^{(n)})) = -\sum_{i=1}^{T_y^{(n)}}\frac{\partial}{\partial\theta}s_{ik} - \mathbb{E}_{v\sim\mathcal{V}}\left[\frac{\partial}{\partial\theta}\exp(s_{iv})\right]. \quad (3)$$

As computational complexity of Equation (3) grows linearly in the size of the vocabulary $\mathcal{V}$, it is often the case that tens of thousands of the most frequent words are included in the vocabulary.

3 What is Behind the Out-of-Vocabulary Problem?

In the experimental results of (Jean et al., 2015), we can see that a larger vocabulary does not bring us the performance improvement when the target language is German while English to French translation benefits from the use of a larger vocabulary. This tells us that a larger vocabulary might not be an effective solution to the OOV problem, at least, when we translate into German. To look into the reasons, let us consider what type of words are treated as OOV words and how often such words appear in the corpus. To be more precise, we collect the most frequent $30,000$ words from a large corpus, i.e., the training set being used in our experiments. Then we calculate the following two scores for each of part of speech (POS) and named entity (NE) tags with respect to OOV words as follows

$$\text{coverage(tag)} = \frac{\text{\# of words with } \textit{tag} \text{ in vocabulary}}{\text{\# of words with } \textit{tag}}, \qquad \text{ratio(tag)} = \frac{\text{\# of words with } \textit{tag}}{\text{\# of words}}.$$

For POS and NE tagging, we use the Stanford NLP suite[1].

The results are summarized in Table 1, where we merge the different tagsets for German and English into common tags to make them comparable. In particular, all different types of adjectives fall into a single *ADJ* tag. The ratio refers to the frequency in the test set, e.g., 20.50% for *NN* means that 20.50% of the words in unseen data are nouns. As can be seen in the table, NEs in all forms, i.e., I-PER, I-ORG and I-LOC, are poorly covered. This is intuitively obvious because a fixed-size vocabulary cannot contain the names of all people, places, or organizations. NEs also appear frequently, i.e., 5% of all unknown words are NEs. The coverage of numbers (*CARD*) is also suboptimal, with the English vocabulary covering 92.01% while the German vocabulary covers only 87.62%. For numbers as well as for NEs this may not be a problem per se because OOV words can be copied from the source sentence using the alignment model.

[1] http://stanfordnlp.github.io/CoreNLP

The biggest gap of coverage of a tag set between English and German can be seen for nouns (*NN*). English features a coverage of 93.25% while German covers only 68.73%. The reason for this is most likely due to an interesting property of the German language: compound words. In German, new words can be created by concatenating two or more words.

Interestingly, it is hard to get 95% of German words covered by a vocabulary of even 1M words in contrast to English. Figure 1 shows coverage of words in the test set as a function of vocabulary size. Unless German compound words are split, coverage of German is lower than that of English by $3 \sim 8\%$ depending on the vocabulary size. Splitting compound words enables to cover German words in the test set as much as English ones as the vocabulary size increases.

4 Dealing with Compound Words

Compounds are words (more precisely, lexemes) consisting of more than one stem. Compounding is found in many languages, including Dutch, Finnish, and German. To build a compound word, two or more words are joined resulting in a longer word. For example, the German compound word *Apfelkuchen* is produced by concatenating *Apfel* (apple) and *Kuchen* (pie). Yet in German, it needs to be explicitly added to the vocabulary while its corresponding term, *apple pie*, is likely to be already included in the vocabulary extracted from a large English corpus. As Germanic languages such as German have a high number of compound words, we explore the methodology of compound splitting for German being both on the source and target side, referred to as *compound splitting* and *compound merging*, respectively.

4.1 Compound Splitting

Compound splitters often employed in the literature generally fall into one of two categories (Fritzinger and Fraser, 2010; Popović et al., 2006): *corpus-driven* and *linguistic*. The former is based on word frequencies calculated on a training corpus, the best split being selected among the candidates. The latter splits words through linguistic analysis e.g., by using a morphological analyzer. Hybrid approaches are possible, and corpus-driven splitters found in practice often contain some additional rules instead of relying solely on corpus statistics. Likewise, while it is possible to split words based solely on the output of a morphological analyzer, a splitter may take word frequencies into account.

In this work, we consider the following two splitters:

1. The compound splitter developed by Weller and Heid (2012), herein referred to as *IMS splitter*, is based on the corpus-driven approach by Koehn and Knight (2003). We compute a frequency list of lemmatized words using TreeTagger (Schmid, 1994; Schmid, 1995) on the WMT14 training data as well as a frequency list of words with respect to their POS tag.[2] These two lists are used to improve the performance of the IMS splitter. Different splitting possibilities are ranked by the geometric mean of the frequencies $(\prod_{i=1}^{n} \text{freq}(p_i))^{\frac{1}{n}}$ where p_i are the parts of the respective splittings and n the number of parts. An exemplary output of this splitter can be found in Table 2, which shows different ways to split a word along with the scores for this particular split.

2. Fritzinger and Fraser (2010) studied compound splitters in terms of their effectiveness in statistical machine translation (SMT) systems and propose a hybrid approach that uses corpus statistics as well as a morphological analyzer. Hence, we will refer to this splitter as *hybrid splitter*. Based on split points provided by the morphological analyzer *Smor*, word frequencies from a large training corpus are consulted. Smor is a finite-state based morphological analyzer concerned with the word formation of German, i.e., inflection, derivation, and compounding. By employing a hybrid approach, only linguistically motivated splittings (provided by Smor) are performed according to the frequency lists on the training corpus, hence reducing the number of incorrect splits. For example, The word *Durchschnittsauto* (*standard car*) is split into *Durchschnitt Auto* instead of *Durch Schnitt Auto*. The word *Durchschnitt* has a one-to-one relation with the English word *average*, but is itself a compound word.

[2] We switched to TreeTagger from the Stanford Tagger because of its substantially faster processing speed.

Table 2: Exemplary output of the IMS splitter.

Compound	Split Compound	Score
kühleinrichtung	kühl_ADJ einrichtung_NN	869.3
	kühlen_V einrichtung_NN	251.4
	kühleinrichtung_NN	6.0
gezeitenkraftwerk	gezeiten_NN kraft_NN werk_NN	984.9
	gezeiten_NN kraftwerk_NN	324.44
	gezeitenkraft_NN werk_NN	243.6
	gezeitenkraftwerk_NN	33.0

Another important aspect when talking about compound splitting is the concept of *filler letters*. When a new word is formed through the use of compounding, sometimes filler letters are inserted between two compounds. For German, these filler letters are usually *s, en, n, er, and es*. For example, the German word *Jahresverpflichtung* is made from the words *Jahr* (year) and *Verpflichtung* (commitment), and between these two words the infix *es* is inserted. The easiest way to deal with this issue is to maintain a list of filler letters and a set of rules for testing whether a compound ends with one of them, in which case these filler letters can be removed. Indeed, the IMS splitter uses exactly this type of rules to provide proper splits. The hybrid splitter does not need an explicit definition of filler letters because this knowledge is already encoded in the morphological analyzer Smor.

For the inverse task of translating from English to German, it is not so clear what filler letters should be used because filler letters are needed for reconstructing a correct compound word from its individual elements.

4.2 Compound Merging

When compounds occur on the target side it is necessary to join individual compound elements together. We can deal with this problem by inserting special syntax into the training corpus that eases the process of compound merging. For example, instead of encoding the word *Autofahrer* (car driver) as *Auto* and *Fahrer*, we can encode it as *Auto @@ Fahrer*. By doing so, we hope the model will produce such connection markers in the decoder, which can then easily be joined during postprocessing. This also solves the problem of filler letters for German being on the target side, because filler letters can be encoded using this special syntax as well. For instance, the word *Kraftverkehrsverband* is made of up the words *Kraft*, *Verkehr*, and *Verband* and can be encoded as *Kraft @@ Verkehr @s@ Verband*.

This process has also been proposed by (Williams et al., 2015) using the hybrid splitter by (Fritzinger and Fraser, 2010). We argue that this special syntax can also be produced by a much simpler corpus-driven splitter such as the IMS splitter. Hence, we modified the IMS splitter to output these filler letters between compound words. This modification applies to all compound splitters simply by comparing the splitter's output with its input and extracting filler letters using a regular expression.

5 Experimental Setup

This section describes the foundations of our experiments. In particular, we address preprocessing, which is an important part of the machine learning pipeline, and the setup of our NMT training.

5.1 Dataset & Preprocessing

Our models were trained on the data provided by the 2014 Workshop on Machine Translation (WMT). Specifically, we used the Europarl v7, Common Crawl, and News Commentary corpora. Our development set corresponds to the data in the *newstest2013* files while the test set is given by *newstest2014*.[3] We use Moses' tokenizer[4] and filter out noisy sentences, written in a different language, using a language detection tool.[5]

[3] We use cleaned test sets.

[4] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

[5] https://github.com/shuyo/language-detection

Since some of the German sentences in the training set were made available before the German orthography reform of 1996, we converted such words to conform to the new spelling rules. For instance, the following rules were used: (muß ⟶ muss), (Phantasie ⟶ Fantasie), (Bestelliste ⟶ Bestellliste), (essentiell ⟶ essenziell) and (geschrieen ⟶ geschrien). Note that this spelling conversion does not affect the development and test sets because no occurrences of words written in the old style were present.

5.2 Training Details

Our NMT implementation is based on the neural network framework `Blocks`[6] (van Merriënboer et al., 2015). While it may be advisable to try different hyperparameters, an elaborate tuning of the RNNs was not practical given that it took about one week to train a single hyperparameter configuration for machine translation on the WMT data set on our hardware. Therefore, we chose the same set of hyperparameters as in (Bahdanau et al., 2015) except for the size of the vocabularies. To be more specific, we set the dimensionality of word vectors to 640, the number of hidden units in GRU-RNN to $1,000$, the maximum sequence length during training to 50, the mini-batch size to 80, and the number of words in both source and target vocabularies to $30,000$. We did not use weight noise regularization or dropout. We also randomly shuffled the training data using a constant seed for all our experiments so that different models are trained on the same order of the data set.

We trained our models on a NVIDIA Tesla K20 and Titan Black GPU. Since our data set contains nearly 4 million instances, a single epoch corresponds to around $48,000$ iterations. Depending on the model, the best score on the development set was usually achieved at around $400,000$ to $500,000$ iterations, or 8 to 10 epochs, or 5 to 7 days. At every checkpoint, $15,000$ iterations in our experiments, we computed the bilingual evaluation understudy (BLEU) score (Papineni et al., 2002), the Metric for Evaluation of Translation with Explicit ORdering (METEOR) (Banerjee and Lavie, 2005) and the translation error rate (TER) (Snover et al., 2006) to monitor the progress of the model on the development set.

6 Experiments

In this section, we present the results for the experiments we conducted with the setup explained in the previous sections. We will first show that German to English translation works considerably better than English to German (Section 6.1), and then try to analyze this observation in more depth (Section 6.2).

6.1 Effect of Compound Splitting and Joining

Table 3 shows the performance of three models on the test set for both translation directions. All our models produce cased outputs. Nonetheless, BLEU scores are given for evaluation performed before and after lowercasing predictions and reference sets.

For English to German, both compound splitters contribute to the improvements of NMT models across all measures except for TER although the gap of the performance between the models is negligible. However, for German to English translation, we can observe huge gains by using the corpus-based compound words splitting method, outperforming the baseline by 2.7 BLEU$^{\text{uncased}}$, 3.3 METEOR, and -0.6 TER. While the use of compound splitters enables to achieve significant performance improvement in terms of BLEU and METEOR, only marginal improvement was observed in TER. This implies that splitting German compound words results in more number of correct n-grams in English whereas it has no significant impact on word ordering. See more details of three metrics in (Birch and Osborne, 2011). Note that German word order can differ from that of English due to separated verb prefixes and long-range movement of verbs in German.

When inspecting the output of both splitters, we noticed that the hybrid/linguistic splitter is more likely to produce a correct splitting. However, the corpus-based splitter provides better results for NMT. This contradiction has also been observed by (Fritzinger and Fraser, 2010) for phrase-based SMT. Corpus-driven methods tend to split more aggressively. For instance, consider the word *überall* which may get split into *über* and *all* by the corpus-driven approach due to these two words appearing very frequently. Conversely, the linguistic splitter is unlikely to split this word.

[6] `https://github.com/mila-udem/blocks`

Table 3: Results of compound split for German words using either a corpus-based splitter or hybrid corpus-based/linguistic-motivated one. Scores in subscripts are on the development set.

Translation Direction	Model	↑ BLEU$^{\text{uncased}}$	↑ BLEU$^{\text{cased}}$	↑ METEOR$^{\text{uncased}}$	↓ TER$^{\text{uncased}}$
English → German	Baseline	$19.21_{19.6}$	$18.79_{19.11}$	$39.40_{39.00}$	$63.40_{61.70}$
	Hybrid Split	$19.34_{19.58}$	$18.95_{19.10}$	$39.90_{39.10}$	$63.70_{62.20}$
	Corpus-based Split	$19.41_{20.22}$	$18.99_{19.78}$	$40.30_{39.80}$	$64.40_{62.10}$
German → English	Baseline	$21.72_{23.30}$	$20.91_{22.36}$	$26.50_{27.90}$	$58.80_{58.70}$
	Hybrid Split	$22.30_{23.55}$	$21.37_{22.50}$	$27.30_{28.40}$	$60.00_{59.30}$
	Corpus-based Split	$24.42_{24.89}$	$23.41_{23.75}$	$29.80_{30.30}$	$58.20_{57.80}$

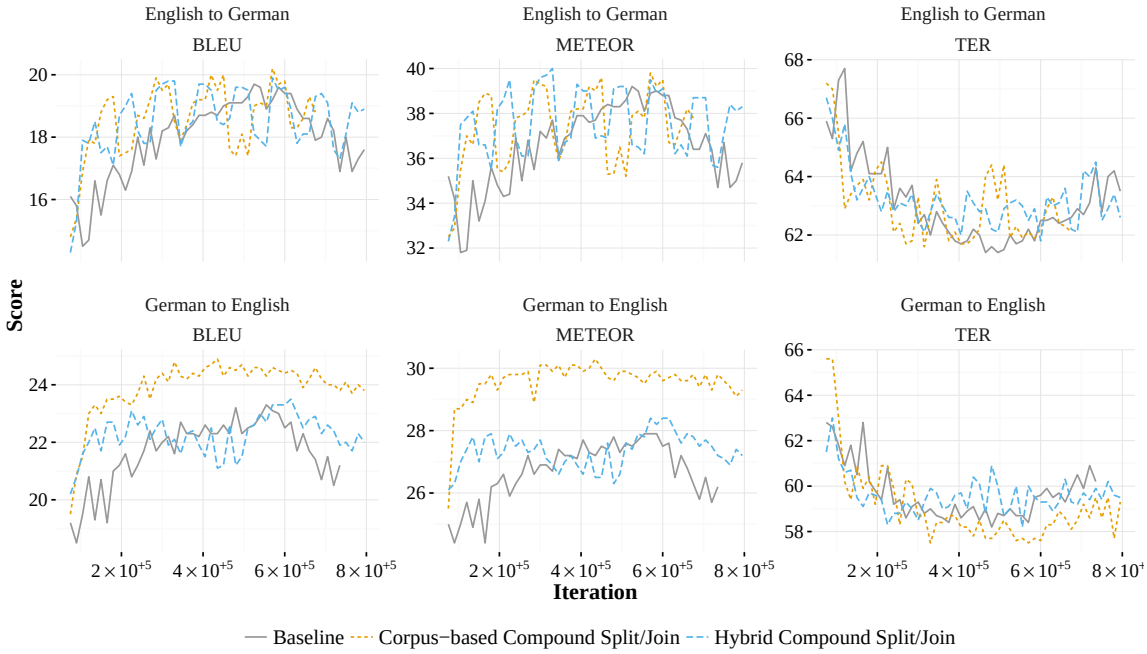

Figure 2: Performance of NMT models handling compound words differently on the development set.

The explanation for a translation system performing better with aggressive splits could be that the system can recover from words that are split too much. This is especially applicable to NMT, where the problem of OOV words leads to many compound words not found in the vocabulary. Hence, the performance hit by unsplit words that may lead to OOV words is larger than the performance hit produced by overaggressive splitting.

We believe that the reason why compound splitting works much better for English than for German is that the vocabulary coverage for English is well above the coverage of German. As noted in Section 3, we found that only 68.73% German nouns are covered when encoding a text corpus using a fixed vocabulary of size $30,000$, whereas English nouns are covered at a ratio of 93.25%. This motivated our experiment that contains a compound splitter. Before we started training a NMT model with compound split data, we first explored the effects of compound splitting in terms of word coverage on the test data. For this analysis, we used the IMS splitter. Our results, which can be seen in Figure 1, confirm that word coverage for German dramatically increases when compound splitting is applied. Another possible reason is that if a compound word at the source side is split into multiple words, the NMT model described in Section 2 is able to learn alignments between English words and multiple subwords of the compound word. We think that this is also related to why the frequency-based vocabulary construction method improves the performance of NMT models (Sennrich et al., 2016).

6.2 Difficulties When Translating into German

When German is on the target side, the model successfully produces the special syntax we used that allows us to concatenate compound words as a postprocessing step (cf. Section 4.2). A translation produced by this model looks as follows:

Source Sentence	The darker the meat, the higher the **pH value**.
Reference Sentence	Je dunkler das Fleisch, desto höher der **ph-Wert**.
Baseline Model	Je dunkler das Fleisch, desto höher der **pH**.
Split/Join Model	Je dunkler das Fleisch, desto höher der **pH @-@ Wert**.
Split/Join Model (merged)	Je dunkler das Fleisch, desto höher der **pH-Wert**.

The translation produced by the model for the phrase *pH value* is *pH @-@ Wert* and can be concatenated by the use of a regular expression. This seems to work even with longer compound words, as shown by the following example:

Source Sentence	A total of four **road safety inspections** were carried out.
Reference Sentence	Insgesamt seien vier **Verkehrsschauen** durchgefhrt worden.
Baseline Model	Insgesamt wurden vier ⟨**UNK**⟩ durchgefhrt.
Split/Join Model	Es wurden insgesamt vier **Straße @n@ Verkehr @s@ Inspektionen** durchgefhrt.
Split/Join Model (merged)	Es wurden insgesamt vier **Straßenverkehrsinspektionen** durchgefhrt.

In this case, the baseline model fails to translate the phrase altogether resulting in an unknown word. For readers not familiar with German, all sentences are correct translations for the source sentence when we disregard the unknown word, i.e., the baseline model and the split model differ in structure, but both are correct, and the split join model is more precisely in translating the word for road safety inspection.

Another interesting observation is that the performance in terms of BLEU fluctuates wildly as splitting is applied. We compared the translation output between iterations showing favorable performance and iterations with a lower BLEU score. We found that sentences were often restructured causing the performance to drop. An example of this restructuring may look as follows:

Source Sentence	However, the new rules put in place will undoubtedly make it more difficult to exercise the right to vote in 2012 .
Reference Sentence	Allerdings werden die neu eingeführten Regeln im Jahr 2012 zweifellos die Ausübung des Wahlrechts erschweren .
Split/Join Model, Iteration 450000	Mit den neuen Regeln wird es zweifellos schwieriger sein, das Wahlrecht im Jahr 2012 auszuüben .
Split/Join Model, Iteration 465000	Die neuen Regeln werden jedoch zweifellos die Ausübung des Wahlrechts für 2012 erschweren .

The system output at iteration $450,000$ in this example receives 35.29 BLEU points, whereas the output at iteration $465,000$ receives 71.43 BLEU points. This discrepancy can be explained with the fact that BLEU as an evaluation measure favors sentences with more n-gram matches. The model's output at iteration $450,000$ receives a 4-gram match for the phrase *zweifellos die Ausübung des Wahlrechts erschweren*, whereas the other model's output does not. Please do note that both outputs are perfect translations of the source sentence and would be evaluated similarly by a human evaluator.

7 Conclusion

In this work, we presented why word-level NMT models have trouble with OOV words and explained why such problems arise. Based on the analysis about the OOV problem, we obtained substantial improvements of 2.7 BLEU points for German to English translations by splitting German compound words on the source side. We also applied compound splitting when German was on the target side by modifying the compound splitter's output to contain special syntax that can later be used to create compound words from their individual components. While we obtained only a modest improvement of 0.2 BLEU, our proposed method is able to merge German words on the target side, allowing us to generate correct target sentences.

Acknowledgements

Calculations on the Lichtenberg high performance computer of the Technische Universität Darmstadt were conducted for this research. The Titan Black GPU used for this research was donated by the NVIDIA Corporation. This work has been supported by the German Institute for Educational Research (DIPF) under the Knowledge Discovery in Scientific Literature (KDSL) program, and the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representation*, San Diego, California.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for machine translation evaluation with improved correlation with human judgments. In *Proceedings of the Association for Computational Linguistics Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, volume 29, pages 65–72.

Alexandra Birch and Miles Osborne. 2011. Reordering metrics for mt. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1027–1035, Portland, Oregon, June. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1693–1703, Berlin, Germany.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380, Baltimore, Maryland.

Fabienne Fritzinger and Alexander Fraser. 2010. How to avoid burning ducks: combining linguistic analysis and corpus statistics for german compound processing. In *Proceedings of the 5th Joint Workshop on Statistical Machine Translation and MetricsMATR*, pages 224–234, Uppsala, Sweden.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1–10, Beijing, China.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 187–193, Budapest, Hungary.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1054–1063, Berlin, Germany.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Beijing, China.

Kishore Papineni, 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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Maja Popović, Daniel Stein, and Hermann Ney. 2006. Statistical machine translation of german compound words. In *Advances in Natural Language Processing*, volume 4139, pages 616–624. Springer.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, volume 12, pages 44–49, Manchester, UK.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *Proceedings of the Association for Computational Linguistics (ACL) Special Interest Group for Linguistic Data and Corpus-Based Approaches to Natural Language Processing (EMNLP)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, Berlin, Germany.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: frameworks for deep learning. *arXiv preprint arXiv:1506.00619*.

Marion Weller and Ulrich Heid. 2012. Analyzing and aligning german compound nouns. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 2395–2400, Istanbul, Turkey.

Philip Williams, Rico Sennrich, Maria Nadejde, Mathias Huck, and Philipp Koehn. 2015. Edinburgh's syntax-based systems at WMT 2015. In *Proceedings of the 10th of the Association for Computational Linguistics Workshop on Statistical Machine Translation*, pages 199–209, Lisbon, Portugal.