Improving Neural Machine Translation by Incorporating Hierarchical Subword Features

Makoto Morishita, Jun Suzuki* and **Masaaki Nagata**
NTT Communication Science Laboratories, NTT Corporation, Japan
{morishita.makoto, nagata.masaaki}@lab.ntt.co.jp
jun.suzuki@ecei.tohoku.ac.jp

Abstract

This paper focuses on subword-based Neural Machine Translation (NMT). We hypothesize that in the NMT model, the appropriate subword units for the following three modules (layers) can differ: (1) the encoder embedding layer, (2) the decoder embedding layer, and (3) the decoder output layer. We find the subword based on Sennrich et al. (2016) has a feature that a large vocabulary is a superset of a small vocabulary and modify the NMT model enables the incorporation of several different subword units in a single embedding layer. We refer these small subword features as hierarchical subword features. To empirically investigate our assumption, we compare the performance of several different subword units and hierarchical subword features for both the encoder and decoder embedding layers. We confirmed that incorporating hierarchical subword features in the encoder consistently improves BLEU scores on the IWSLT evaluation datasets.

Title and Abstract in Japanese

階層的部分単語特徴を用いたニューラル機械翻訳

本稿では、部分単語 (subword) を用いたニューラル機械翻訳 (Neural Machine Translation, NMT) に着目する。NMT モデルでは、エンコーダの埋め込み層、デコーダの埋め込み層およびデコーダの出力層の 3 箇所で部分単語が用いられるが、それぞれの層で適切な部分単語単位は異なるという仮説を立てた。我々は、Sennrich et al. (2016) に基づく部分単語は、大きな語彙集合が小さい語彙集合を必ず包含するという特徴を利用して、複数の異なる部分単語列を同時に一つの埋め込み層として扱えるよう NMT モデルを改良する。以降、この小さな語彙集合特徴を階層的部分単語特徴と呼ぶ。本仮説を検証するために、様々な部分単語単位や階層的部分単語特徴をエンコーダ・デコーダの埋め込み層に適用して、その精度の変化を確認する。IWSLT 評価セットを用いた実験により、エンコーダ側で階層的な部分単語を用いたモデルは BLEU スコアが一貫して向上することが確認できた。

1 Introduction

The approach of end-to-end Neural Machine Translation (NMT) continues to make rapid progress. A simple encoder-decoder model was proposed by Sutskever et al. (2014), and an attentional mechanism was added to better exploit the encoder-side information (Luong et al., 2015; Bahdanau et al., 2015). Compared to traditional Statistical Machine Translation (SMT), NMT has relatively simple architecture, which only uses a large single neural network, but its accuracy surpasses SMT (Junczys-Dowmunt et al., 2016).

A conventional NMT uses a limited vocabulary and a decoder generates a "word" in the vocabulary at each time step, but a problem occurs when it encounters an out-of-vocabulary word. Since NMT cannot correctly encode and generate such out-of-vocabulary words, the task performance is degraded. To solve this problem, Sennrich et al. (2016) proposed a method that expresses a word by combining "subwords."

*His current affiliation is Tohoku University.

Proceedings of the 27th International Conference on Computational Linguistics, pages 618–629
Santa Fe, New Mexico, USA, August 20-26, 2018.

A subword is a fraction of a word, determined by Byte Pair Encoding (BPE) operations. By the BPE operation, a word that appears frequently can be one unit, and rare or uncommon words can be expressed by the combination of several subword units. Thus we can express any words by the combination of small subword vocabularies to alleviate the out-of-vocabulary problem. Several similar works exist to make subword units (e.g., (Schuster and Nakajima, 2012; Kudo, 2018)), but in the following, we denote the units segmented by BPE as subword units unless otherwise noted.

The primary reason why we use subword units is to generate rare or unknown words on the decoder side. In other words, our purpose is to change the vocabulary at the decoder output layer into subwords. Once we decide the vocabulary in the decoder output layer, it is natural to use the same vocabulary in the decoder embedding layer. We also use subword units in the encoder side to maintain consistency with the decoder. As described, NMT has three layers that are related to subwords: the encoder embedding layer, the decoder embedding layer, and the decoder output layer. However, we generally use the same operations to make subword units.

We hypothesize that the optimal subword units can be different among these three layers. Since these layers play different roles in the model, the subword units should be determined based on each role. To validate this hypothesis, we modify the model to simultaneously deal with several different subword units.

We focus on the property that the large subword vocabulary is always a superset of the small subword vocabulary. By taking advantage of this, we propose the model uses these small subword vocabularies as additional features of an embedding layer. We name these small subword vocabulary features as hierarchical subword features. We simply use the sum of the embeddings of each hierarchical subword features to represent each embedding. This simple approach is GPU friendly and does not increase the computational time.

We empirically investigate our assumption and find that incorporating several different subword units for encoder embedding layers consistently improves the BLEU scores on the IWSLT 2012, 2013, and 2014 evaluation datasets.

2 Neural Machine Translation with Subword Units

Among many options for a model architecture of NMT models, our baseline's model architecture was introduced in Luong et al. (2015) with a global attention mechanism and a bi-directional encoder (Bahdanau et al., 2015).

2.1 Formulation

In general, the NMT model receives an input sentence and returns a corresponding (translated) output sentence. Here, to concisely explain the NMT model, its input and output are both sequences of one-hot vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ that respectively correspond to input and output sentences. This conversion can be performed straightforwardly without loss of generality since each token (word) has a one-to-one correspondence to a one-hot vector.

Let $\mathcal{V}_s$ and $\mathcal{V}_t$ respectively represent the vocabulary sizes of the input and the output. Let $\boldsymbol{x}_i \in \{0,1\}^{V_s}$ represent the one-hot vector of the i-th token in $\boldsymbol{X}$. Similarly, let $\boldsymbol{y}_j \in \{0,1\}^{V_t}$ represent the one-hot vector of the j-th token in $\boldsymbol{Y}$. We introduce notation $\boldsymbol{x}_{1:I}$ to represent a list of one-hot vectors, i.e., $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I)$, as a short notation where I represents the length (the number of instances) of the list. Then the NMT model approximates the following conditional probability:

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{j=1}^{J+1} p(\boldsymbol{y}_j|\boldsymbol{y}_{0:j-1}, \boldsymbol{X}), \tag{1}$$

where $\boldsymbol{y}_0$ is a one-hot vector of a special begin-of-sentence (BOS) token and $\boldsymbol{y}_{J+1}$ is a one-hot vector of a special end-of-sentence (EOS) token. Moreover, $\boldsymbol{X} = \boldsymbol{x}_{1:I}$ and $\boldsymbol{Y} = \boldsymbol{y}_{0:J+1}$.

Our baseline NMT model consists of three primary components (modules): encoder, attention, and decoder. The following briefly explains these three components. Hereafter, we assume that the number

of tokens in the input sentence is I, the number of tokens in the output sentence is J, the dimensions of the embedding vectors are D, and the dimensions of the hidden vectors are H.

Encoder: The encoder generates a list of hidden vectors $\boldsymbol{h}_{1:I}$ given an input sequence of one-hot vectors $\boldsymbol{x}_{1:I}$. Let $\boldsymbol{E} \in \mathbb{R}^{D \times |\mathcal{V}_s|}$ represent an (encoder) embedding matrix. Then $\texttt{Enc}(\cdot)$, which denotes a function that returns a list of encoded vectors $\boldsymbol{h}_{1:I}$, is calculated:

$$\boldsymbol{h}_{1:I} = \texttt{Enc}(\boldsymbol{e}_{1:I}), \quad \text{where} \quad \boldsymbol{e}_i = \boldsymbol{E}\boldsymbol{x}_i \quad \text{for all } i. \tag{2}$$

Finally, the encoder outputs $\boldsymbol{h}_{1:I}$.

Decoder and attention mechanism: The decoder estimates the probability of the output sentence given the encoded information generated by the encoder: $\boldsymbol{h}_{1:I}$. The attention mechanism allows the decoder to directly incorporate $\boldsymbol{h}_{1:I}$ at each decoder time step j.

Let $\boldsymbol{F} \in \mathbb{R}^{D \times |\mathcal{V}_t|}$ represent an (decoder) embedding matrix. Let $\texttt{Dec\_Attn}(\cdot)$ be a function that returns the final hidden vector at decoder time step j, namely, $\boldsymbol{z}_j$, which is calculated based on $\boldsymbol{f}_j$, $\boldsymbol{z}_{j-1}$, and $\boldsymbol{h}_{1:I}$ for all j:

$$\boldsymbol{z}_j = \texttt{Dec\_Attn}\big(\boldsymbol{z}_{j-1}, \boldsymbol{f}_j, \boldsymbol{h}_{1:I}\big), \quad \text{where} \quad \boldsymbol{f}_j = \boldsymbol{F}\boldsymbol{y}_{j-1}. \tag{3}$$

Here we assume that both $\boldsymbol{y}_j$ and $\boldsymbol{z}_j$ are zero-vectors if $j = 0$.

Then let $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}_t| \times H}$ and $\boldsymbol{b} \in \mathbb{R}^{|\mathcal{V}_t|}$ be a weight matrix and a bias term in the decoder's output layer. Finally, the decoder calculates the probability of $\boldsymbol{y}_j$ at each time step j, which is $p(\boldsymbol{y}_j | \boldsymbol{y}_{0:j-1}, \boldsymbol{X})$ from Eq. 1:

$$p(\boldsymbol{y}_j | \boldsymbol{y}_{0:j-1}, \boldsymbol{X}) = \frac{\exp(\boldsymbol{o}_j) \cdot \boldsymbol{y}_j}{\exp(\boldsymbol{o}_j) \cdot \mathbf{1}}, \quad \text{where} \quad \boldsymbol{o}_j = \boldsymbol{W}\boldsymbol{z}_j + \boldsymbol{b}, \tag{4}$$

where $\mathbf{1}$ is a vector whose elements are all 1.

In the generation (test) phase, we generally use a K-best beam search to generate output sentences with the (approximated) K-highest probability given input sequence $\boldsymbol{X}$.

2.2 Subword Units Based on Byte-Pair Encoding

Several approaches have been proposed to obtain a set of subword units based on statistics, e.g., (Schuster and Nakajima, 2012; Sennrich et al., 2016). The scheme based on Byte-Pair Encoding (BPE) (Sennrich et al., 2016) is one of the most frequently used methods in current NMT researches. Following this trend, this paper focuses only on a method based on BPE to obtain a set of subword units and refers to Sennrich et al. (2016)'s method as SubW_{BPE} to distinguish it from others for clarity.

The following briefly describes the SubW_{BPE} procedure for building a set of subword units given a set of training data. SubW_{BPE} first splits the input sentences into character units and then combines the two frequently appearing consecutive (character or subword) units into one subword unit. SubW_{BPE} repeats this merge operation predefined m times. For splitting sentences into obtained subword units, we straightforwardly apply merge operations in the obtained order of the above procedure.

Here we revisit several interesting properties of SubW_{BPE}. For example, generated subword units become identical to the character units if and only if we set the number of merge operations m to zero ($m = 0$). All the subword units always recover and match the original word units if we set m to ∞ ($m = \infty$). These two properties imply that SubW_{BPE} naturally involves the methods using either character or word units in a single unified framework in terms of using subword units. Therefore, using SubW_{BPE}, we are not required to distinguish the methods of using character or word units since they are respectively just a special case of SubW_{BPE} with certain hyperparameters $m = 0$ and $m = \infty$. From this perspective, this paper does not explicitly distinguish among character, subword, and word units and treats all of them as subword units.

Another interesting property of SubW_{BPE} is that every subword unit obtained by $m = m'$ can be represented as a series of subword units obtained by $m = m''$ if $m' > m''$. This property is easily provable. SubW_{BPE} consists only of a *merge operation* of two consecutive subwords during the subword

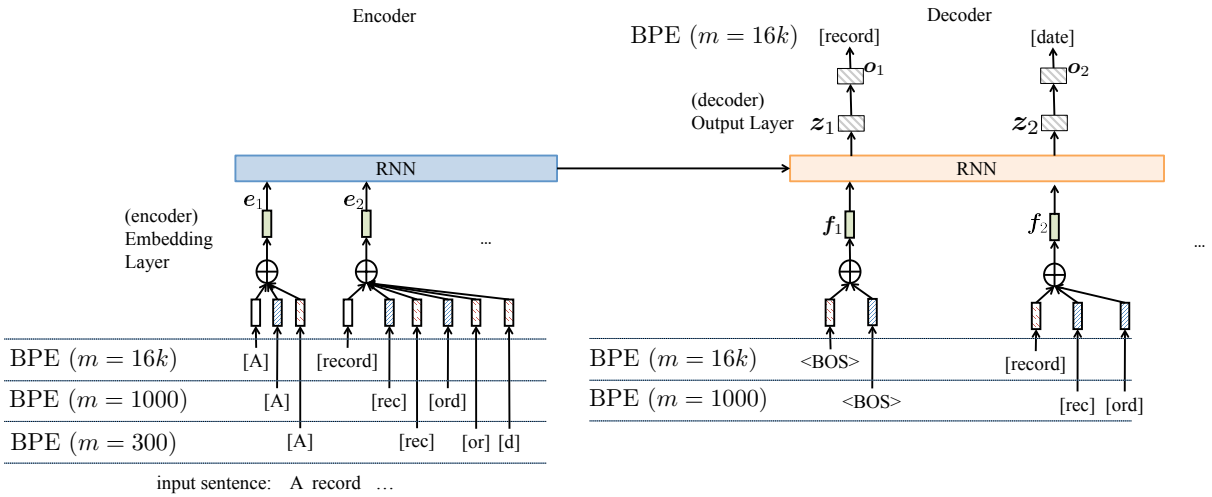

Figure 1: Overview of hierarchical subword features

unit construction, and thus, subword units with m are always identical to a subword with $m-1$ except the merged subwords as the m-th merge operation. Therefore, there is a relation where a subword unit obtained by a larger m is always a concatenation of (several) subword units obtained by a smaller m. This relation is always satisfied by any m' and m'' pairs, indicating a hierarchical *subword in subword* structure for all m from 1 to ∞.

Generally, the number of merge operations (m) is a hyperparameter that is empirically derived. Recent NMT researches usually set this m within a range from 1,000 to 100,000: apparently never less than 1,000 or over 100,000. This is because a set of subword units obtained with few merge operations resembles character units, and therefore the sentence becomes too long to process[1] On the other hand, since a set of subword units obtained with relatively large merge operations becomes nearly identical to the original word units, it lacks the advantage of using subword units. Hence, it is intuitively reasonable to set m in a range from 1,000 to 100,000.

3 Hierarchical Subword Features

Fig. 1 shows an overview of our proposed method. In it, we extend the encoder's and the decoder's embedding layers by modifying our model to work with several subwords units at once.

To formally explain our modification, we first introduce distinct Q encoder embedding matrices and R decoder embedding matrices. Let $\boldsymbol{E}_q$ represent the q-th encoder embedding matrix, where $q \in \{1, \ldots, Q\}$. Similarly, let $\boldsymbol{F}_r$ represent the r-th decoder embedding matrix, where $r \in \{1, \ldots, R\}$. Then we modify the operations to obtain encoder and decoder embedding vectors, which are shown respectively in Eqs. 2 and 3. For the encoder embedding vectors, we introduce the following operator:

$$\boldsymbol{e}_i = \sum_q \boldsymbol{E}_q \phi_q(\boldsymbol{x}_i), \tag{5}$$

where $\phi_q(\boldsymbol{x}_i)$ is a mapping function that returns a binary vector that corresponds to $\boldsymbol{x}_i$. For the decoder embedding vectors, we obtain $\boldsymbol{f}_j$ by the following equation:

$$\boldsymbol{f}_j = \sum_r \boldsymbol{F}_r \psi_r(\boldsymbol{y}_{j-1}), \tag{6}$$

where $\psi_r(\boldsymbol{y}_{j-1})$ is a mapping function that returns a binary vector that corresponds to previously estimated result $\boldsymbol{y}_{j-1}$. For example, if we get `record` as an estimation result of BPE(m =16k), mapping

[1]NMT lacks the ability to translate longer sentences (Koehn and Knowles, 2017).

	DE-EN		FR-EN	
	Tokens	Sentences	Tokens	Sentences
train	3,496,036	189,318	3,800,613	208,323
tst2012 (development)	30,900	1,700	21,653	1,124
tst2013 (test)	21,037	993	21,894	1,024
tst2014 (test)	24,950	1,305	24,950	1,305

Table 1: Cleaned corpora statistics on IWSLT datasets. Number of tokens is English side.

Configurations	Values	Configurations	Values
Embedding dimension D	512	Optimizer	SGD
Hidden dimension H	512	Initial learning rate	1.0
Attention dimension	512	Gradient clipping	5.0
Encoder layer	2	Dropout rate	0.3
Decoder layer	2	Mini-batch size	128 sent

Table 2: Model and optimization configurations

function $\psi_r(\boldsymbol{y}_{j-1})$ returns a binary vector that corresponds to the subwords with smaller merge operations (e.g., `rec` and `ord`). As we described in Section 2.2, every subword unit obtained with $m = m'$ can be represented as a series of subword units obtained with $m = m''$ if $m'' < m'$, and the series of subword units are uniquely determined. Thus, our modification can be interpreted as adding features of smaller subword units, which were derived from previously estimated output $\boldsymbol{y}_{j-1}$. We refer to our method that incorporates smaller subword features as *hierarchical subword features*.

Note that we use different embeddings for each hierarchical subword feature[2]. This means that NMT can learn different features for each hierarchical subword.

The hierarchical subword features slightly increased the number of model parameters (see Section 4.2.1 for more detail). However, increasing the memory requirement and runtime is limited, which allows us to run almost the same speed and memory requirement as the baseline system[3].

We can simultaneously use several subword features. In Fig. 1, we use both BPE (m =1k) and BPE (m =300) for the encoder side. By adding several subword features, the model can use more information with which we expect to improve the task performance.

4 Experiments

4.1 Setup

In this paper, we focused on from/to English (EN) to/from French (FR), German (DE) translations. We carried out our experiments on the IWSLT evaluation campaign dataset (Cettolo et al., 2012), which is based on a TED talk that has been translated into several languages. We used the IWSLT 2016 training set for the training models, tst2012 as the development set, and tst2013 and tst2014 as the test sets. For preprocessing, we used the Moses tokenizer[4] and the truecaser[5]. For the training set, we removed sentences over 50 words to clean the corpus. Table 1 shows the cleaned corpora statistics on the IWSLT datasets. To split words into subwords, we used the scripts[6] provided by Sennrich et al. (2016).

As an NMT framework, we used almost the same structure as Luong et al. (2015), except for our proposed embedding layers. Detailed NMT configurations are shown in Table 2. We set the initial

[2]We also tested using the same embeddings. However, we found through preliminary experiments that it is more effective to use different embeddings.

[3]The number of model parameters and size of the memory requirement largely depend on the vocabulary size. However, we use a smaller m (e.g., 1k or 300) for the hierarchical subword features, and since these features only affect the embedding layer, its effect on computational cost is limited.

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

[5]`https://github.com/moses-smt/mosesdecoder/blob/master/scripts/recaser/truecase.perl`

[6]`https://github.com/rsennrich/subword-nmt`

	Encoder		Decoder		
	Unit	Feature	Unit	Feature	Description
(a)	16k	16k	16k	16k	baseline, conventional setting
(b)	1k	1k	16k	16k	smaller vocabulary setting for encoder side
(c)	300	300	16k	16k	smaller vocabulary setting for encoder side
(d)	16k	16k, 1k	16k	16k	(a) + Encoder side BPE 1k feature
(e)	16k	16k, 300	16k	16k	(a) + Encoder side BPE 300 feature
(f)	16k	16k, 1k, 300	16k	16k	(a) + Encoder side BPE 1k, 300 features
(g)	∞	∞	16k	16k	substituting Encoder side BPE 16k in (a) to word
(h)	∞	∞, 16k	16k	16k	(g) + Encoder side BPE 16k feature
(i)	∞	∞, 1k	16k	16k	(g) + Encoder side BPE 1k feature
(j)	∞	∞, 300	16k	16k	(g) + Encoder side BPE 300 feature
(k)	∞	∞, 1k, 300	16k	16k	(g) + Encoder side BPE 1k, 300 features
(l)	16k	16k	16k	16k, 1k	(a) + Decoder side BPE 1k feature
(m)	16k	16k	16k	16k, 300	(a) + Decoder side BPE 300 feature
(n)	16k	16k	16k	16k, 1k, 300	(a) + Decoder side BPE 1k, 300 features
(o)	16k	16k, 1k, 300	16k	16k, 1k, 300	(f) + (n)
(p)	∞	∞, 1k, 300	16k	16k, 1k, 300	(k) + (n)

Table 3: Compared experimental settings

learning rate to 1.0, but after 30 epochs we multiplied it by 0.8 for every epoch and continued training until 40 epochs. For decoding, we performed a beam search with a beam size of 20. To prevent the model from outputting short sentences, we applied the length normalization technique by dividing the negative log-likelihood by the sentence length (Cromieres et al., 2016; Morishita et al., 2017). As evaluation metrics, we used case-sensitive[7] BLEU scores (Papineni et al., 2002) using `multi-bleu.perl`[8].

To fix the experimental settings, we carried out a preliminary analysis to find the relation between the sentence length and the vocabulary size ($\simeq$ the number of BPE merge operations). Fig. 2 shows the results on the English sentences of the IWSLT 2016 German-English training set. When we reduce the vocabulary size, the average sentence length rapidly increases. Unfortunately, longer sentences require more computational cost and are time-consuming. Thus in our experiments, we set the baseline system's vocabulary size to 16,000, which balances the sentence length without affecting the advantage of the subwords.

The experimental settings are compared in Table 3. Our experiments answer the following questions:

- Does the hierarchical subword feature improve the model?
- Which part of the model should we use it? The encoder side, decoder side or both?
- How does it affect the translation results?

4.2 Results

Tables 4 and 5 show the experimental results with various word units and hierarchical subword features. All the scores are averages of four independently trained models. We used different parameter initialization and random seeds to train them. The score differences between the baseline system (a) and the proposed system are shown in the brackets.

(b) and (c) show some improvements using smaller subword units. However, as mentioned in Section 4.1 and shown in Fig. 2, these units are too small and lengthen the sentences. Thus computational time is also extended (see Table 6).

Then we added hierarchical subword features to the encoder side. From (d), (e), and (f), we confirmed that these features improved the model. System (f) uses both BPE (m=1k) and (m=300)

[7]For a reference, we used a true-cased test set.

[8]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

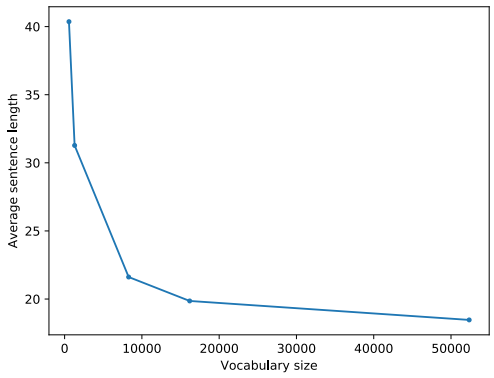

Figure 2: Relation between vocabulary amount ($\simeq$ number of BPE merge operations) and average sentence length in DE-EN training set (EN side). As vocabulary increases, length is shortened.

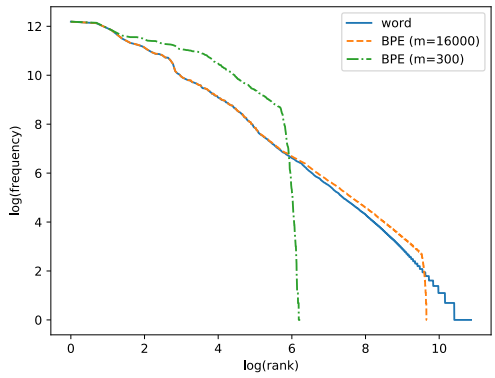

Figure 3: Relation between (sub-)word frequency and its rank in DE-EN training set (EN side). Both axes are in log scale.

	DE-EN						EN-DE					
	tst2012		tst2013		tst2014		tst2012		tst2013		tst2014	
(a)	31.64		33.68		29.18		26.24		28.22		24.17	
(b)	32.23	(+0.59)	34.24	(+0.56)	29.52	(+0.34)	26.18	(−0.06)	28.09	(−0.13)	24.46	(+0.29)
(c)	32.48	(+0.84)	34.60	(+0.92)	29.91	(+0.73)	26.39	(+0.15)	28.75	(+0.53)	24.70	(+0.53)
(d)	32.18	(+0.54)	34.49	(+0.81)	29.60	(+0.42)	26.50	(+0.26)	28.92	(+0.70)	24.74	(+0.57)
(e)	32.15	(+0.51)	34.83	(+1.16)	29.69	(+0.51)	26.78	(+0.54)	28.90	(+0.68)	24.79	(+0.61)
(f)	32.45	(+0.82)	34.67	(+0.99)	29.92	(+0.74)	27.07	(+0.83)	29.10	(+0.88)	24.88	(+0.70)
(g)	30.37	(−1.27)	32.58	(−1.10)	27.53	(−1.65)	25.86	(−0.38)	28.28	(+0.06)	24.28	(+0.11)
(h)	30.79	(−0.85)	33.14	(−0.54)	28.18	(−1.00)	26.06	(−0.18)	28.30	(+0.08)	24.31	(+0.14)
(i)	32.39	(+0.75)	34.76	(+1.09)	29.93	(+0.75)	26.61	(+0.37)	29.13	(+0.91)	24.73	(+0.56)
(j)	32.43	(+0.80)	34.63	(+0.95)	29.78	(+0.60)	26.90	(+0.66)	29.25	(+1.03)	25.15	(+0.98)
(k)	32.71	(+1.07)	34.71	(+1.03)	30.06	(+0.88)	26.99	(+0.75)	29.16	(+0.94)	**25.28**	(+1.11)
(l)	31.62	(−0.02)	33.60	(−0.08)	29.11	(−0.07)	26.11	(−0.13)	28.38	(+0.16)	24.22	(+0.05)
(m)	31.55	(−0.08)	33.65	(−0.03)	29.21	(+0.03)	26.17	(−0.07)	28.26	(+0.04)	24.15	(−0.03)
(n)	31.65	(+0.01)	33.51	(−0.17)	29.00	(−0.18)	25.93	(−0.31)	28.37	(+0.15)	24.13	(−0.04)
(o)	**32.84**	(+1.20)	34.76	(+1.09)	29.99	(+0.81)	**27.27**	(+1.03)	29.14	(+0.92)	24.97	(+0.79)
(p)	32.80	(+1.17)	**34.95**	(+1.27)	**30.11**	(+0.93)	27.11	(+0.87)	**29.64**	(+1.42)	25.20	(+1.03)

Table 4: BLEU scores on IWSLT German-to-English and English-to-German experiments. All scores are *averages* of four independently trained models.

as hierarchical subword features and shows more improvement than using a single feature. Since these models use BPE(m =16k) as a unit, the computational cost is almost the same as the baseline system (a).

In (d), (e), and (f), we used BPE (m =16k) as a unit, but it is more natural to use "word" (BPE ($m = \infty$)) as a unit. For the comparison, we did experiments from word to BPE translation without hierarchical subword features (g). As we expected, that step degraded the accuracy more than the baseline with BPE. This is because its vocabulary contains many rare words, and thus it is difficult to train these word embeddings well. However, by adding hierarchical subword features ((h) to (k)), the accuracy improved at the same level as the system with BPE units. Hierarchical subword features helped the model correctly encode the rare words and improved the accuracy.

For system (h), we saw no improvement, perhaps because large subword units (e.g., BPE (m=16k)) seem too similar to the word units, so it did not help the model very much. Fig. 3 shows the relation

	FR-EN						EN-FR					
	tst2012		tst2013		tst2014		tst2012		tst2013		tst2014	
(a)	42.35		39.61		36.79		43.65		40.09		37.32	
(b)	42.84	(+0.49)	39.53	(−0.08)	36.86	(+0.07)	43.72	(+0.07)	40.43	(+0.34)	37.69	(+0.37)
(c)	43.54	(+1.19)	39.42	(−0.19)	37.16	(+0.37)	43.86	(+0.21)	40.11	(+0.02)	37.54	(+0.22)
(d)	43.13	(+0.78)	39.93	(+0.32)	37.29	(+0.50)	44.59	(+0.94)	40.75	(+0.67)	37.82	(+0.51)
(e)	43.18	(+0.83)	39.42	(−0.19)	37.34	(+0.55)	44.76	(+1.11)	41.25	(+1.16)	38.29	(+0.97)
(f)	43.60	(+1.25)	40.01	(+0.40)	37.42	(+0.62)	45.07	(+1.42)	41.15	(+1.06)	38.50	(+1.18)
(g)	41.70	(−0.65)	38.36	(−1.26)	35.83	(−0.96)	43.13	(−0.52)	39.38	(−0.70)	36.54	(−0.78)
(h)	42.04	(−0.31)	38.56	(−1.05)	36.19	(−0.60)	42.88	(−0.77)	39.40	(−0.69)	36.54	(−0.78)
(i)	43.31	(+0.96)	40.13	(+0.52)	37.17	(+0.38)	44.84	(+1.19)	41.07	(+0.98)	38.11	(+0.79)
(j)	43.34	(+0.99)	40.27	(+0.65)	37.44	(+0.65)	45.01	(+1.36)	41.43	(+1.35)	**38.61**	(+1.29)
(k)	**43.82**	(+1.47)	**40.38**	(+0.77)	**37.94**	(+1.15)	45.32	(+1.67)	**41.58**	(+1.50)	38.51	(+1.20)
(l)	42.47	(+0.12)	39.19	(−0.42)	36.63	(−0.16)	43.86	(+0.21)	40.22	(+0.13)	37.25	(−0.07)
(m)	42.34	(−0.01)	39.52	(−0.09)	36.96	(+0.17)	43.79	(+0.14)	39.85	(−0.24)	37.14	(−0.18)
(n)	42.55	(+0.20)	39.09	(−0.52)	36.87	(+0.08)	43.54	(−0.11)	39.84	(−0.24)	37.20	(−0.12)
(o)	43.62	(+1.27)	40.12	(+0.51)	37.73	(+0.94)	45.22	(+1.57)	41.32	(+1.23)	37.98	(+0.66)
(p)	43.63	(+1.28)	39.93	(+0.31)	37.22	(+0.43)	**45.43**	(+1.78)	41.50	(+1.42)	38.34	(+1.03)

Table 5: BLEU scores on IWSLT French-to-English and English-to-French experiments. All scores are *averages* of four independently trained models.

	Model parameters	Training times for an epoch
(a)	38.9M	1050s
(b)	31.3M	1075s
(c)	30.9M	1101s
(d)	39.7M	981s
(e)	39.2M	992s
(f)	40.0M	1002s
(g)	88.8M	1053s
(h)	97.1M	1069s
(i)	89.5M	1029s
(j)	89.1M	976s
(k)	89.8M	1015s
(l)	39.7M	987s
(m)	39.2M	967s
(n)	39.9M	1004s
(o)	41.1M	1019s
(p)	90.8M	1083s

Table 6: Number of model parameters and required training times for an epoch (DE-EN). We used a single NVIDIA GeForce GTX 1080 Ti GPU for training. Required training time might vary due to server condition.

between the (sub-)word frequency and its rank on the English sentences of the German-English training set. The word's graph follows Zipf's law, and thus these word embeddings that appear a few times in the training set are relatively hard to train. The graph shows that BPE (m =16k) is almost the same as the word, because the frequently appearing subword pairs are connected until a subword to be a word. On the other hand, since each subword in BPE ($m = 300$) appears in the training set more frequently, its embedding layer is trained well. Perhaps the number of merge operations for the hierarchical subword features should be smaller than the one we commonly use. Our method can use both small and large subword features at once, thus we can cherry pick an advantage of small subword with maintaining the sentence length.

For systems (l), (m), and (n), we also added features to the decoder side, but we did not find as much improvement as the encoder side. A possible reason for this observation is that our method works

	DE-EN						EN-DE					
	tst2012		tst2013		tst2014		tst2012		tst2013		tst2014	
(a)	34.98		37.52		32.54		28.67		30.73		26.51	
(b)	35.03	(+0.05)	37.08	(−0.44)	31.97	(−0.57)	28.79	(+0.12)	30.67	(−0.06)	26.52	(+0.01)
(c)	35.27	(+0.29)	37.20	(−0.32)	32.60	(+0.06)	28.67	(+0.00)	30.94	(+0.21)	26.66	(+0.15)
(d)	35.44	(+0.46)	37.62	(+0.10)	**33.28**	(+0.74)	28.87	(+0.20)	31.21	(+0.48)	27.00	(+0.49)
(e)	35.02	(+0.04)	37.53	(+0.01)	32.99	(+0.45)	29.13	(+0.46)	31.12	(+0.39)	26.72	(+0.21)
(f)	35.46	(+0.48)	**37.88**	(+0.36)	33.07	(+0.53)	29.04	(+0.37)	30.99	(+0.26)	26.94	(+0.43)
(g)	32.49	(−2.49)	34.98	(−2.54)	29.24	(−3.30)	28.13	(−0.54)	30.59	(−0.14)	25.95	(−0.56)
(h)	34.59	(−0.39)	37.59	(+0.07)	32.23	(−0.31)	29.11	(+0.44)	31.20	(+0.47)	**27.01**	(+0.50)
(i)	35.08	(+0.10)	37.64	(+0.12)	32.62	(+0.08)	29.01	(+0.34)	31.23	(+0.50)	26.71	(+0.20)
(j)	35.12	(+0.14)	37.80	(+0.28)	32.69	(+0.15)	28.90	(+0.23)	31.46	(+0.73)	26.74	(+0.23)
(k)	35.08	(+0.10)	37.86	(+0.34)	32.69	(+0.15)	**29.26**	(+0.59)	31.17	(+0.44)	26.73	(+0.22)
(l)	34.90	(−0.08)	37.29	(−0.23)	32.71	(+0.17)	28.77	(+0.10)	30.89	(+0.16)	26.41	(−0.10)
(m)	34.62	(−0.36)	37.61	(+0.09)	32.77	(+0.23)	28.84	(+0.17)	30.35	(−0.38)	26.11	(−0.40)
(n)	34.92	(−0.06)	36.79	(−0.73)	32.10	(−0.44)	28.71	(+0.04)	31.21	(+0.48)	26.37	(−0.14)
(o)	**35.64**	(+0.66)	37.85	(+0.33)	32.99	(+0.45)	29.24	(+0.57)	31.12	(+0.39)	26.67	(+0.16)
(p)	35.25	(+0.27)	37.78	(+0.26)	32.60	(+0.06)	29.03	(+0.36)	**31.62**	(+0.89)	26.72	(+0.21)

Table 7: BLEU scores on IWSLT German-to-English and English-to-German experiments. All scores are *ensembles* of four independently trained models.

as a regularizer of the model and might degrade the decoder's language modeling ability: in other words, its ability to predict the next token given previous tokens.

Even though (o) and (p) systems showed slight improvements from (f) and (k), such insignificant improvements suggest that the usefulness of (o) and (p) is limited.

Our results suggest the following conclusions: (1) add hierarchical subword features to the encoder side but not to the decoder side and (2) use fewer merge operations, e.g., $m = 300$ and $m =$1k.

4.2.1 Number of Parameters and Required Training Time

Table 6 shows the number of parameters and required training times per epoch. The number of model parameters significantly increases if we add *word*-level features ((g), (h), (i), (j), (k), and (g)). In contrast, adding *subword*-level features does not significantly increase the number of parameters.

We also checked the required training times for each setting. The training time with hierarchical subword features is comparable to the baseline NMT. These results revealed that our methods do not require further computational costs and can be easily applied to any existing systems.

4.2.2 Model Ensembling Results

Tables 7 and 8 show the BLEU scores of ensembling four independently trained models. Hierarchical subword features consistently improved the BLEU scores even for ensembling. This means that our method can be applied to highly tuned systems such as the one submitted to WMT.

4.2.3 Example of Improved Translation

Table 9 shows an example of an improved translation from French to English. The input includes a rare combination of two words, "Britney Spears", which is a proper noun that is hard to translate. Table 10 shows an example of how the words "Britney Spears" were segmented into subwords. These words have been split into small bits of subwords. The subwords are slightly different based on the number of merge operations.

On the one hand, the baseline system with BPE 16k for both the encoder and decoder side cannot correctly translate the two words. On the other hand, our proposed system with hierarchical subword features did correctly translate them. One significant reason is that the embedding layer of subwords with large merge operations is not trained well, as described in Section 4.2. In contrast, our proposed model can make use of both large and small features for correct translations of such rare words.

	FR-EN						EN-FR					
	tst2012		tst2013		tst2014		tst2012		tst2013		tst2014	
(a)	47.46		43.72		40.48		46.13		42.86		39.79	
(b)	46.07	(−1.39)	41.94	(−1.78)	39.62	(−0.86)	47.19	(+1.06)	**43.57**	(+0.71)	**40.87**	(+1.08)
(c)	46.47	(−0.99)	42.38	(−1.34)	39.51	(−0.97)	46.97	(+0.84)	42.94	(+0.08)	40.71	(+0.92)
(d)	**48.72**	(+1.26)	43.59	(−0.13)	40.98	(+0.50)	46.91	(+0.78)	43.02	(+0.16)	40.07	(+0.28)
(e)	48.16	(+0.70)	43.86	(+0.14)	**41.53**	(+1.05)	46.41	(+0.28)	42.66	(−0.20)	40.38	(+0.59)
(f)	48.18	(+0.72)	43.96	(+0.24)	41.10	(+0.62)	**47.40**	(+1.27)	43.30	(+0.44)	40.37	(+0.58)
(g)	47.61	(+0.15)	42.50	(−1.22)	40.00	(−0.48)	44.88	(−1.25)	41.85	(−1.01)	38.53	(−1.26)
(h)	45.95	(−1.51)	42.22	(−1.50)	39.54	(−0.94)	47.51	(+1.38)	43.54	(+0.68)	40.39	(+0.60)
(i)	48.52	(+1.06)	44.24	(+0.52)	40.67	(+0.19)	46.79	(+0.66)	43.32	(+0.46)	39.82	(+0.03)
(j)	48.61	(+1.15)	43.85	(+0.13)	41.17	(+0.69)	46.24	(+0.11)	42.85	(−0.01)	39.89	(+0.10)
(k)	48.49	(+1.03)	**44.33**	(+0.61)	40.92	(+0.44)	46.76	(+0.63)	42.90	(+0.04)	40.18	(+0.39)
(l)	48.47	(+1.01)	43.99	(+0.27)	40.43	(−0.05)	46.14	(+0.01)	42.86	(+0.00)	39.67	(−0.12)
(m)	48.11	(+0.65)	43.32	(−0.40)	40.68	(+0.20)	46.10	(−0.03)	42.49	(−0.37)	39.92	(+0.13)
(n)	47.94	(+0.48)	43.25	(−0.47)	40.83	(+0.35)	46.62	(+0.49)	42.77	(−0.09)	39.77	(−0.02)
(o)	48.38	(+0.92)	44.09	(+0.37)	40.90	(+0.42)	47.14	(+1.01)	42.98	(+0.12)	40.68	(+0.89)
(p)	48.24	(+0.78)	44.32	(+0.60)	41.00	(+0.52)	46.86	(+0.73)	42.76	(−0.10)	39.96	(+0.17)

Table 8: BLEU scores on IWSLT French-to-English and English-to-French experiments. All scores are *ensembles* of four independently trained models.

Input	J'ai répondu, "Je ne suis pas Britney Spears, mais tu peux peut-être me l'apprendre à moi.
Reference	I was like, "Well I'm not Britney Spears, but maybe you could teach me.
(a) Baseline	I said, "I'm not British Speney Spears, but maybe you can teach me.
(k) Proposed	I said, "I'm not Britney Spears, but maybe you can teach me.

Table 9: Example translation from French to English. Proposed method correctly translated rare words: "Britney Spears."

Merge operations	Subword segmentation
16k	Bri t ney S pe ars
1k	B ri t n e y S pe ars
300	B ri t n e y S p e ar s

Table 10: Example segmentation of "Britney Spears"

5 Related Work

Sennrich and Haddow (2016) added linguistic features to NMT embedding layer and achieved significant improvement. They modified the embedding layer to exploit several features, such as part-of-speech tags and syntactic dependency labels. This method resembles our work in terms of providing more information to the embedding layer. However, to use these linguistic features, since we need to prepare a morphological analyzer and/or a dependency parser, applicable languages are limited. In our method, BPE features are language independent and applicable to all languages.

Our method can also be interpreted as a regularizer to the embedding layer. Recently, Kudo (2018) proposed a subword regularization method that uses different subword segmentation based on its segmentation probability. This method increases NMT's robustness to noise and segmentation errors. Their experiments showed that the subword regularization method is significantly effective when testing accuracy with a different dataset than the training set. This means that their method is effective with open-domain settings. Our method might have a similar tendency, but we will investigate in future work. It might be interesting to verify the effect of combining our method and the subword regularization method.

Several studies incorporated the Recurrent Neural Network (RNN) or the Convolutional Neural Network (CNN) into the embedding layer for encoding character-, subword-, or morpheme-level informa-

tion (Luong and Manning, 2016; Lee et al., 2017). Comparing with these approaches, our work has a significant advantage in terms of the fast computation since our method increases negligible computational cost as clarified in Section 4.2.1.

6 Conclusion

In this paper, we focused on neural machine translation with subword units and experimented with hierarchical subword features. Our experiments confirmed that adding hierarchical subword features to the encoder side consistently improved the BLEU scores. Our method, which is quite simple and easy to adapt to any models that use subwords as a unit (such as text summarization and language modeling), has the potential to be a de-facto standard in the future. Our codes and scripts are available for reproduction and further experiments[9].

In this paper, we just experimented with an RNN-based NMT, even though recently several new NMT architectures have been proposed, including an attentional-based (Vaswani et al., 2017) and a CNN-based NMT (Gehring et al., 2017). As future work, we want to try our method with these new NMT models and see whether it is effective. Future work will also apply hierarchical subword features to a larger dataset.

Acknowledgements

We thank three anonymous reviewers for their insightful comments.

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 Representations (ICLR).

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT3: web inventory of transcribed and translated talks. In Proceedings of the 16th Annual Conference of the European Association for Machine Translation (EAMT), pages 261–268.

Fabien Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto university participation to WAT 2016. In Proceedings of the 3rd Workshop on Asian Translation (WAT), pages 166–174.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In Proceedings of the 34th International Conference on Machine Learning (ICML), pages 1243–1252.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? a case study on 30 translation directions. In Proceedings of the International Workshop on Spoken Language Translation (IWSLT).

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In Proceedings of the 1st Workshop on Neural Machine Translation (WNMT), pages 28–39.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL).

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. Transactions of the Association for Computational Linguistics (TACL), 5:365–378.

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 (ACL), pages 1054–1063.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1412–1421.

[9] https://github.com/nttcslab-nlp/hierarchical_subword

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2017. NTT neural machine translation systems at WAT 2017. In Proceedings of the 4th Workshop on Asian Translation (WAT), pages 89–94.

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 (ACL), pages 311–318.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 5149–5152.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In Proceedings of the 1st Conference on Machine Translation (WMT), pages 83–91.

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 (ACL), pages 1715–1725.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS), pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS), pages 6000–6010.