General Paper

Sub-Subword N-Gram Features for Subword-Level Neural Machine Translation

Ander Martinez†, Katsuhito Sudoh† and Yuji Matsumoto†,††

Neural machine translation (NMT) systems often use subword segmentation to limit vocabulary sizes. This type of segmentation is particularly useful for morphologically complex languages because their vocabularies can grow prohibitively large. This method can also replace infrequent tokens with more frequent subwords. Fine segmentation with short subword units has been shown to produce better results for smaller training datasets. Character-level NMT, which can be considered as an extreme case of subword segmentation in which each subword consists of a single character, can provide enhanced transliteration results, but also tends to produce grammatical errors. We propose a novel approach to this problem that combines subword-level segmentation with character-level information in the form of character n-gram features to construct embedding matrices and softmax output projections for a standard encoder-decoder model. We use a custom algorithm to select a small number of effective binary character n-gram features. Through four sets of experiments, we demonstrate the advantages of the proposed approach for processing resource-limited language pairs. Our proposed approach yields better performance in terms of BLEU score compared to subword- and character-based baseline methods under low-resource conditions. In particular, the proposed approach increases the vocabulary size for small training datasets without reducing translation quality.

Key Words: Feature Selection, Low-resource, Neural Machine Translation, N-gram, Word Segmentation

1 Introduction

Neural machine translation (NMT) has made remarkable progress in recent years (Sutskever et al. 2014; Bahdanau et al. 2014; Vaswani et al. 2017) and has become a popular approach for general machine translation tasks.

NMT systems take a sequence of source language tokens as inputs and predict a sequence of target language tokens as corresponding translation outputs. The granularity of such input and output tokens can be smaller than words, such as subwords (Sennrich et al. 2015). Replacing

† Nara Institute of Science and Technology
†† Riken Center for Advanced Intelligence Project
Martinez et al. Sub-Subword N-Gram Features for Subword-Level Neural Machine Translation

words with sequences of subwords has become a popular method. Subword segmentation methods, such as byte-pair encoding (BPE), replace infrequent words with more frequent subword segments to limit vocabulary sizes. Limiting vocabulary sizes is particularly important for morphologically complex languages because their vocabularies may contain hundreds of thousands of entries. The optimal size for a vocabulary depends on the particular training data. As indicated by Denkowski and Neubig (2017), NMT models trained on smaller corpora tend to perform better with smaller BPE vocabularies, because subwords with very low frequencies are avoided.

However, BPE segmentation does not consider the context of a word or its morphemes. Subwords with similar surface forms are likely to contain the same morphemes, but if these subwords are represented by unique IDs, their similarities are not considered.

Character-level NMT (Chung et al. 2016; Lee et al. 2016), can be regarded as an extreme case of subword segmentation in which each subword corresponds to a single character. Character-level systems can perform better transliteration results than subword-level systems, as shown by Sennrich (2017).

Sennrich (2017) compared the character-level model proposed by Lee et al. (2016) and the BPE-to-character model proposed by Chung et al. (2016) to their own BPE-only model (Sennrich et al. 2016a). They concluded that character-level decoders “perform worse at modelling morphosyntactic agreement, where information needs to be carried over long distances.”

In a more recent paper, Cherry et al. (2018) demonstrated that character-level models with large capacities (large numbers of layers in their encoders and decoders) can outperform BPE when their training datasets are very large (in their experiments, they considered 39.9 million sentence pairs). However, their model suffers from a disproportionate reduction in speed.

In this paper, we explore a novel training method for NMT models that allows us to utilize the character n-gram information of each subword to learn corresponding embeddings. This method is described in detail in Section 3. Our proposed models operate at the subword level and make use of short n-gram features within subwords. In this paper, we refer to the character n-gram features of subwords as sub-subword features.

We propose a subword-level NMT training method that utilizes sub-subword n-gram information to regularize subword embeddings and the output layer. Sub-subword-level features are selected from all n-grams in the subword vocabulary using a custom feature selection algorithm. The set of selected sub-subword features can unambiguously identify every subword in the vocabulary. Sub-subword binary features are fed through a multilayer feed-forward network to produce subword embeddings. Because subword embeddings are produced exclusively from constant sub-subword features, the subword embeddings can be pre-computed once the model has
been trained. Therefore, our method does not need to modify baseline architectures or the numbers of parameters in trained models.

Another method that works well on low-resource language pairs is back-translation (BT) (Sennrich et al. 2016b; Poncelas et al. 2018; Edunov et al. 2018). BT is a simple method that uses an NMT model to translate sentences in a target language into source language sentences, allowing one to synthesize a parallel corpus. We explored how our proposed model can be combined with BT.

Through four sets of experiments, we determined the optimal architecture for applying sub-subword features to subword-level NMT. We determined that this method works particularly well with small training sets. Our main contributions can be summarized as follows:

• We propose an NMT model training approach combining subword segmentation to limit vocabulary sizes with n-gram features to construct subword embeddings using a standard multilayer feed-forward network (Subsection 3.1).
• We use a feature selection algorithm to select a small number of character n-gram features. Selected features should uniquely identify every word in the vocabulary (Subsection 3.2).
• We demonstrate that a standard multilayer feed-forward network works better than a self-attention mechanism (Subsection 4.2).
• We demonstrate that constructing subword embeddings strictly from n-gram features works better than using n-gram features to complement standard embeddings (Subsection 4.2).
• We demonstrate that our method facilitates large vocabulary sizes with small training datasets, thereby improving translation results (Subsection 4.3).
• We demonstrate that our method works well for small training datasets (Subsection 4.4).
• We explore how the proposed method can be combined with BT (Subsection 4.5).

This paper consists of five sections. The Introduction (1) is followed by a Preliminaries section (2). Section 3 describes the proposed approach. Four sets of experiments are detailed in Section 4. Finally, Section 5 presents a summary of our findings and discusses future work.

2 Preliminaries

2.1 Neural Machine Translation

NMT models are used to translate sequences of tokens, such as words, in a source language into corresponding sequences of tokens in a target language. The sets of all distinct tokens in the source and target language sentences are called the source and target vocabularies, respectively.
NMT models operate based on closed vocabularies. The tokens in each vocabulary are assigned unique identification numbers.

The input tokens of a source sentence $x$ are represented as a sequence of one-hot vectors of the corresponding tokens $x := (x_1, \ldots, x_{T_x})$, where $T_x$ is the number of tokens in the sentence. To output a probability distribution $p_t$ for a token $y_t$, the previous output tokens $y := (y_1, \ldots, y_{t-1})$, as well as the input $x$, are conditioned. These tokens are then projected into dense vectors $e_i^{(x)}$ and $e_j^{(y)}$ using matrices $E_x$ and $E_y$, which are often called embedding matrices. Typical NMT models use two embedding matrices, namely $E_x$ for source sentence words and $E_y$ for representing words in the target sentence. When using embeddings of $d$ dimensions and vocabularies of sizes $V_x$ and $V_y$, the embedding matrices $E_x$ and $E_y$ are in $\mathbb{R}^{V_x \times d}$ and $\mathbb{R}^{V_y \times d}$, respectively. Such models output a probability distribution $p_t$ by projecting a dense vector of size $d$ onto a vector of the same size as the output vocabulary using the softmax function. The embedding matrices, output layer matrix $W_y$, and bias $b_y$ are often trained as conventional weight parameters.

$$
e_i^{(x)} := E_x x_i,$$  
$$
e_j^{(y)} := E_y y_j,$$  
$$q_t := \text{NMT} \{(e^{(x)}_{0 \le i \le T_x}), \{e^{(y)}_{0 \le j < t}\}\},$$  
$$p_t := \text{Softmax}(W_y q_t + b_y),$$  
$$\text{Softmax}(z)_k := \frac{e^{z_k}}{\sum_l e^{z_l}},$$

where the function $\text{NMT}$ represents a model that, given an input embedding sequence $e^{(x)}$ and previous translation tokens’ embedding sequence $\{e^{(y)}_{0 \le j < t}\}$, approximates a dense vector $q_t$ corresponding to the next token in the translation.

The technique called weight tying (Press and Wolf 2017) consists in defining $W_y$ as the transpose of $E_y$. If both the source language and target language are represented by the same set of tokens, they can share an embedding matrix, resulting in $E_x = E_y = W_y^\top$.

### 2.2 Subword Segmentation

As mentioned in Section 1, word-level vocabularies can be very large, particularly for morphologically complex languages. Subword segmentation is used to reduce vocabulary sizes as an alternative to selecting the top-$N$ most frequently used words.

The number of different tokens can be reduced by splitting words into smaller subword units. Subword segmentation increases the length of input and output sequences, and smaller vocabularies result in longer sequences. The optimal vocabulary size depends on the training data,
where smaller datasets typically favor smaller vocabularies.

One very popular subword segmentation method is BPE (Sennrich et al. 2015). The BPE subword vocabulary starts with the set of all individual characters in the training data. The final characters in words are considered to be distinct from their counterparts in other locations. A vocabulary is augmented until it has the desired size by merging frequent subword pairs to form new subwords.

There are various approaches to word segmentation in the literature that are similar to BPE. One such approach is the WordPieceModel (Schuster and Nakajima 2012). Instead of merging subwords based on their frequency, this model merges subwords to optimize a likelihood value for a language model. Different merging policies were explored in (Wu and Zhao 2018), including frequency, accessor variety and descriptor length gain. Another approach to word segmentation is the unigram language model (Kudo 2018), which can produce multiple segmentation candidates. In this model, the subword vocabulary is selected to optimize the probabilities of words by considering the probability of one word to be the product of the unigram probabilities of its subwords.

Once data have been segmented into subword units, they are treated as independent tokens in the manner described in Section 2.1. The word embedding matrix can then be considered as a subword embedding matrix containing subword features. The information regarding which characters comprise a token is ignored. The potential benefits of taking advantage of this information represent the main motivation for the use of sub-subword features.

### 2.3 Character n-gram Features

Using n-gram features to generate word vectors is not a new concept: Wieting et al. (2016) used n-gram count vectors to represent words for different natural language processing (NLP) tasks. Bojanowski et al. (2017) trained word vectors on a skipgram language model by representing each word as the sum of the vectors representing its n-grams. This method takes word-boundary symbols into consideration. They considered n-gram sizes ranging from three to six characters. Because representing each n-gram using a unique vector can consume excessive memory, n-grams are grouped into buckets using a hash function, where they share vector representations with other n-grams in the same bucket. Joulin et al. (2017) used the same approach to develop a model called fastText for text classification. N-gram vectors can complement word vectors, because words are also unique n-grams. Zhang et al. (2019) used subword segmentation to complement word-level vectors with subword-level vectors for various NLP tasks that did not include NMT. They compared various subword segmentation methods introduced by Wu and
Martinez et al. Sub-Subword N-Gram Features for Subword-Level Neural Machine Translation

Zhao (2018). Morishita et al. (2018) reinterpreted BPE subwords from smaller vocabulary configurations as features of subwords in a larger vocabulary. The embedding vector for a given subword was represented as the sum of its feature vectors. Based on memory constraints, they only used sub-subword features for input and output embeddings, and not for the final output layer. Takase et al. (2019) explored the use of n-gram features of a fixed size for generating embeddings and output layer weights for language modeling. They also conducted experiments on NMT using trigram features to produce embeddings. In contrast to other approaches, they used a self-attention mechanism to calculate a weighted sum of feature embeddings.

We briefly describe the model proposed by Takase et al. (2019), as we will compare our proposed model to it. For a more detailed description, please refer to the original work.

Takase et al. (2019) incorporated embeddings inferred from fixed length n-grams (such as bigrams) into standard embeddings by replacing (1) and (2) with the following two equations:

\[ e^{(x)}_i := E^x x_i + c(x_i, \theta_c^{(x)}), \]
\[ e^{(y)}_j := E^y y_j + c(y_j, \theta_c^{(y)}), \]

where \( c(\omega, \theta_c) \) is a neural network with a parameter set \( \theta_c \) that produces dense vector inferred from the n-gram features of word \( \omega \).

The n-gram features augment word embeddings without replacing them. To produce \( c(x_i, \theta_c^{(x)}) \) and \( c(y_j, \theta_c^{(y)}) \) from n-gram features, Takase et al. used what they called multidimensional self-attention. If the number of n-gram features of a specific word \( \omega \) is given by \( I(\omega) \in \mathbb{N} \), then

\[ c(\omega, \{W_c, S\}) := \sum_{i=1}^{I(\omega)} g_i \odot s_i^{(\omega)}, \]
\[ S^{(\omega)} := (s_1^{(\omega)}, \ldots, s_{I(\omega)}^{(\omega)}), \]
\[ g_i := [\text{RowSoftmax}(W_c S^{(\omega)})]_i, \]

where \( \odot \) denotes element-wise product of vectors, \([\cdot]_i\) is the i-th column of a given matrix and RowSoftmax is a softmax function applied row wise. The matrices \( S^{(\omega)} \in \mathbb{R}^{D_c \times I(\omega)} \) represent the embeddings of the n-gram features in the word \( \omega \). The dimension of the embeddings is \( D_c \). These n-gram feature embeddings and the square matrix \( W_c \in \mathbb{R}^{D_c \times D_c} \) are trainable parameters, that can differ between input and output layers.
3 Proposed Method

3.1 Training Method

In this study, we explored different approaches utilizing n-gram features to represent translation tokens instead of learning their representations as embedding matrix parameters or using n-gram features to augment embeddings. Because we used BPE subwords to limit vocabulary sizes in our experiments, we will refer to the translation tokens as “subwords” and to their n-gram features as “sub-subword features.”

We denote $V_x$ and $V_y$ as the vocabulary sizes, and $d_x$ and $d_y$ as the numbers of n-gram features for the input and output languages, respectively. During training, our model learns to produce embeddings and output matrices from two sparse binary feature matrices $F_x \in \mathbb{R}^{V_x \times d_x}$ and $F_y \in \mathbb{R}^{V_y \times d_y}$ for the source and target language vocabularies, respectively. Each element in the feature matrices indicates whether an n-gram feature is included in a subword. When the input and output vocabularies are the same, such as when training BPE subwords jointly, the feature matrices $F_x$ and $F_y$ are equal. We will refer to this part of the model as the feature-to-embedding (FTE) network. Unlike the method presented by Takase et al. (2019), we use a simple feed-forward network.

Our proposed approach does not train feature matrices. Equations (1), (2), and (4) are replaced with the following equations:

$$ e^{(x)}_i := \phi_x(F_x) x_i, $$

$$ e^{(y)}_j := \phi_y(F_y) y_j, $$

$$ p_t := \text{Softmax}(\phi_o(F_y) q_t + \phi_b(F_y)). $$

The parameter weights for the feed-forward network, namely $\phi_x$, $\phi_y$, and $\phi_o$, are updated during training. If weight tying is applied, then

$$ \phi_o(F_y) = \phi_y(F_y)^T. $$

Figure 1 presents a diagram of the training model using weight tying.

Once the model is trained, we revert to equations (1), (2), and (4) by pre-computing the values of $E_x$, $E_y$, $W_y$, and $b_y$ using the following equations:

$$ E_x := \phi_x(F_x), $$

$$ E_y := \phi_y(F_y), $$

$$ W_y := \phi_b(F_y), $$

88
Fig. 1 Simple diagram showing an encoder-decoder model using features to produce embeddings. The inputs and output are depicted with dashed borders and neural subnetworks have rounded corners. The model depicted in this diagram uses weight tying. The output bias vector $b_y$ is not shown for simplicity. On top of the baseline model, the boxes containing the n-gram feature matrices $F_x$ and $F_y$, and the feed-forward networks $\phi_x$ and $\phi_y$ have been added. $E_x$ and $E_y$ are not trainable parameters.

$$b_y := \phi_b(F_y).$$ (18)

After these calculations are completed, the FTE weights are no longer necessary and can be discarded. The resulting model is the same size as the base model.

Our goal is to compare different architectures and n-gram features and to determine their utility. As shown later in Section 4, we determined that a three-layer feed-forward network yields the best results among the compared architectures. We use a rectifier activation function between layers.

Using all of the fixed-width n-grams can result in very large feature matrices. In practice, when using all available features, we can only train models on bigram features based on memory constraints. To be able to use longer n-grams, we use a custom feature selection algorithm. We extract all n-grams for every possible $n$ and select a final subset using the algorithm described in Section 3.2.

### 3.2 Feature Selection

The goal of the proposed algorithm is to select a small number of features that unambiguously represent a given vocabulary. As an example, consider a vocabulary that contains the subwords $ana$ and $anana$. When only using bigram features, both subwords are represented as $\{\text{-a, an, na, a$}\}$. In order to disambiguate these two subwords, the trigram feature $nan$ may be selected. The algorithm selects the feature that disambiguates the maximum number of
subword occurrences.

Given a large set of potential character n-gram features, the proposed feature selection algorithm selects a small subset. The initial feature-set contains features for all n-grams contained in the subword vocabulary, where n ranges from one up to the maximum subword length in the vocabulary.

Consider a vocabulary $V$ containing all subwords in a training dataset. The proposed algorithm selects one character n-gram feature at a time to construct a binary decision tree. Selection can be limited to a given number of features $M$ by stopping early. The selected feature partitions the vocabulary into positive and negative subsets. The subwords in the former subset include the selected feature and those in the latter subset do not. For every subset, we select the subword with the highest unigram frequency and regard it as reachable. If there is a tie, one of the top-frequency subwords is selected randomly to be regarded as reachable. The rest of the subwords in the vocabulary are considered to be unreachable. The optimization objective is to maximize the total unigram probabilities of reachable subwords. The features that best partition the vocabulary in terms of objective are selected. The subword embeddings will be derived strictly from the selected character n-gram features.

Algorithm 1 presents the process described above in the form of pseudo-code. At each step, there is a set of partitions $G$ and candidate features. Considering all combinations of these partitions and candidate features is very expensive. Instead of considering all partitions in $G$, we only consider a subset $\Phi(G)$ containing the partitions with highest probabilities $p_g(g)$, defined in Algorithm 1, summing up to a predefined threshold $T_g$. When selecting a candidate n-gram feature $f$, we also consider a fixed number ($N_\Psi$), which is denoted as $\Psi(G)$. $\Psi(G)$ consists of the features with the top-$N_\Psi$ highest $\Omega(f, G)$ values. $\Omega(f, G)$ ranks the relevance of the features with respect to their frequency within the partitions, and it is defined as follows:

\[
\Omega(f, G) := \sum_{g \in \Phi(G)} \omega(f, g)p_g(g),
\]

\[
\omega(f, g) := 1 - \left| 1 - 2 \frac{p_f(f, g)}{p_g(g)} \right|,
\]

\[
p_f(f, g) := \sum_{\{v \in g \mid f \in Q(v)\}} p_v(v),
\]

where $Q(v)$ represents the n-gram features that appear in $v$.

These optimizations are optional and can be disabled by defining $\Phi(G) = G$ and $\Psi(G)$ as the set of all the features that have not been selected.

To illustrate the selections made by the proposed algorithm, next we present some statistics
**Data:** \( V \), a set of subword tokens; \( v \in V \);
\( Q(v) \), n-gram features that appear in \( v \);
\( p_v(v) \), unigram probability of \( v \); and
\( M \), maximum number of n-gram features

**Result:** \( F_d \), selected feature-list where \( d \leq M \)

Let:

\[
\begin{align*}
    p_g(g) &:= \sum_{v \in g} p_v(v) \\
    p_{\text{max}}(g) &:= \max_{v \in g} p_v(v)/p_g(g) \\
    p_{\text{mix}}(f, g) &:= \\
    &\begin{cases} 
        (p_{\text{max}}(\{v | v \in g, f \in Q(v)\})) \\
        + p_{\text{max}}(\{v | v \in g, f \notin Q(v)\}) 
    \end{cases} \\
    \text{choose feature}(G) &:= \\
    \arg \max_{f \in \Psi(G)} \sum_{g \in \Phi(G)} p_{\text{mix}}(f, g)p_g(g)
\end{align*}
\]

\( F_0 \leftarrow [\] \\
\( G_0 \leftarrow \{V\} \)

for \( i \leftarrow 1 \) to \( M \) do

\[
    f_i \leftarrow \text{choose feature}(G_{i-1}) \\
    F_i \leftarrow F_{i-1} \cdot [f_i] \\
    G^{(1)}_i \leftarrow \{\{v | v \in g, f_i \in Q(v)\} | g \in G_{i-1}\} \\
    G^{(0)}_i \leftarrow \{\{v | v \in g, f_i \notin Q(v)\} | g \in G_{i-1}\} \\
    G_i \leftarrow \{g | G^{(0)}_i \cup G^{(1)}_i \} \mid |g| > 1\}
\]

if \( |G_i| = 0 \) then break

end

\( d \leftarrow i \)

return \( F_d, d \)

**Algorithm 1** Feature selection algorithm. \( \Psi \) and \( \Phi \), defined in the text, are used to prune the search space. The notation \([\] \) represents an empty sequence, \([f_i] \) represents a sequence of a single element \( f_i \), and the operator \( \cdot \) denotes sequence concatenation.

for an English-Turkish dataset when using a vocabulary size of approximately 64,000 words and \( T_g = 0.95 \). The dataset contained 2,399 bigrams and 14,933 trigrams. The number of trigram features was too large to train a model using the proposed approach on a single GPU. The number of features selected by the feature selection algorithm was 598, out of which 28.8% were unigrams, 43.5% were bigrams, 14.5% were trigrams, 8% were four-grams, 3.8% were five-grams, and the remaining 1.4% were longer n-grams. Other datasets yielded similar statistics. Table 1 shows some features of more than two characters.
4 Experiments and Analysis

To investigate the effectiveness of the proposed approach, we conducted four sets of experiments. The experiments in Subsection 4.2 explain the choices made by the proposed approach in comparison to other approaches. Subsection 4.3 demonstrates how the proposed approach relates to the chosen vocabulary size and Subsection 4.4 demonstrates how it relates to the training dataset size. Finally, Subsection 4.5 demonstrates how the proposed approach can be combined with BT.

4.1 Experimental Setup

We ran experiments targeting five languages: English (En), Turkish (Tr), German (De), French (Fr) and Finnish (Fi). The source language was Turkish for the models targeting English. For the rest of the models, the source language was English. The Tr-En and En-tr models were trained using the same dataset. Table 2 shows the number of sentences of each dataset. These datasets were sampled to make smaller training sets to simulate a low-resource setting.

The training dataset for the English-Turkish translation task comes from (Tiedemann 2012) and contains 205,000 parallel sentences. The German and French data were distributed as part of WMT-2014 (Bojar et al. 2014; Koehn 2005) and the Turkish and Finnish data as part of WMT-2017 (Bojar et al. 2017; Koehn 2005; Tiedemann 2012). For each language pair, the
corresponding newstest was used to calculate the displayed BLEU scores (Papineni et al. 2002). The reported BLEU scores are case-sensitive and computed by sacrebleu (Post 2018). It should be noted that the BLEU scores reported by sacrebleu are not computed in the same way as those reported by Vaswani et al. (2017).

The baseline model was a transformer “base” (Vaswani et al. 2017) with eight attention heads, six layers in the encoder, and six layers in the decoder with embeddings of size 512 and 2,048 units in the hidden layers. The hidden layers for producing embeddings (FTE) contained 3,000 units, except when indicated otherwise. This implementation was based on Sockeye (Hieber et al. 2017). Each training mini batch contained approximately 4,096 target words. All models were trained on a single GeForce GTX 1080 Ti GPU.

In our experiments, the input and target languages shared a subword vocabulary and sub-subword features. All related weights were shared too. As a result, $E_x = E_y = W_y = E$ and $F_x = F_y = F$.

The significance testing was done using bootstrap resampling (Koehn 2004). We consider improvements with P-value smaller than 0.05 to be significant.

4.2 Approach Comparison Experiments

In this subsection, we compare different approaches to determine the best architecture. All models were trained using a vocabulary size of approximately 32,000 subwords. All training datasets contained approximately 200,000 sentences. Excluding the Turkish-English pairs, the training data were sub-sampled from larger datasets.

Table 3 lists the results for different models divided into three sections. Section A compares the feed-forward models ($l$-FF, where $l \in [1, 4]$) to the self-attention ($att$) approach proposed by Takase et al. (2019). The $1$-FF approach is equivalent to summing feature embeddings, similar to some of the models introduced in Section 1 (Wieting et al. 2016; Morishita et al. 2018). One can see that the best results in Section A are provided by the three-layer feed-forward models. Adding extra layers does not result in any significant gains in terms of BLEU scores.

Section B in Table 3 compares the proposed feature selection algorithm to a naive approach of selecting the features with highest frequency. Both model sets use a similar number of features, but the proposed selection algorithm results in scores comparable to those achieved when using all bigrams, whereas the naive approach yields poor results. When using the frequency-selected bigrams approach, many subwords contain non-unique representations. As an example, if the features “po”, “pa”, “oi” and “ai” were not selected, the words point and paint would have the same representation: {“^p”, “in”, “nt”, “t$”}. In such cases, the subwords are indistinguishable
Table 3 Different approaches compared in terms of BLEU score results. The models starting with “2-gram” used all known bigrams as features, while those starting with “selection” use the proposed feature selection algorithm. The 1-FF models use 1-layered feed-forward networks to produce embeddings, while the those containing “att” use the approach proposed by Takase et al. (2019). For the +comp models, the embeddings produced from the features are used to complement the traditional embeddings through addition. It should be noted that the numbers of parameters for all models are the same after training. Best scores excluding subword regularization are shown in bold.

and the output layer will always output the one with highest unigram probability.

While the cited approaches (Wieting et al. 2016; Morishita et al. 2018; Takase et al. 2019) use n-gram features to complement standard trainable embeddings, our proposed approach avoids training subword embeddings directly. Section C reveals that for small training datasets, using features to complement standard embeddings is sub-optimal. For example, consider the results for models 2-gram 3-FF and 2-gram 3-FF +comp, where the later complements n-gram features with subword features. The results are consistently better for the model without subword embeddings. We believe this occurs because without subword embeddings, models are forced to infer subword features from n-gram features, resulting in better generalization. In contrast, when subword embeddings are available subword features can be represented directly based on subword embeddings without using n-gram features, resulting in poor generalization and the learning of training data biases.

Section D in Table 3 explores the subword regularization technique proposed by Kudo (2018). Their approach produced good results with significant improvements with respect to the baseline, in accordance with their report. The sampling hyperparameters were $\alpha = 0.2$ and $l = \infty$. 

|       | Tr $\rightarrow$ En | En $\rightarrow$ Tr | En $\rightarrow$ De | En $\rightarrow$ Fr | En $\rightarrow$ Fi |
|-------|---------------------|----------------------|----------------------|----------------------|----------------------|
| Baseline | 16.7               | 13.1                 | 18.0                 | 26.8                 | 12.1                 |
| A 2-gram 1-FF | 16.2               | 13.0                 | 17.3                 | 25.8                 | 12.6                 |
| A 2-gram 2-FF | 17.3               | 13.9                 | 18.7                 | 27.8                 | 13.8                 |
| A 2-gram 3-FF | 17.6 $\mathbf{14.3}$ | 18.9 $\mathbf{28.2}$ | 14.1                 |                     |                      |
| A 2-gram 4-FF | 17.5 $\mathbf{14.3}$ | 18.8                 | 28.1                 | 14.1                 |                      |
| A 2-gram att | 17.4               | 13.4                 | 18.0                 | 27.1                 | 13.3                 |
| B selection 3-FF | $\mathbf{17.8}$ | 14.1                | $\mathbf{18.9}$ | 28.0 $\mathbf{14.2}$ |                     |
| B 2-gram-freq 3-FF | 6.2               | 3.5                  | 3.7                  | 6.8                  | 2.4                  |
| C 2-gram 3-FF +comp | 16.2               | 13.4                 | 18.2                 | 27.2                 | 12.6                 |
| C selection 3-FF +comp | 16.5               | 13.4                 | 18.4                 | 27.1                 | 12.8                 |
| D Regularization | 18.5               | 15.3                 | 19.2                 | 27.4                 | 15.6                 |
| D Regularization + selection 3-FF | 18.7               | 15.6                 | 19.3                 | 27.3                 | 15.7                 |
Martinez et al. Sub-Subword N-Gram Features for Subword-Level Neural Machine Translation

Their method obtained significantly better BLEU scores than our proposed approach for Turkish-English, English-Turkish, and English-Finnish. The subword regularization model also obtained a higher BLEU score for English-German but the gain (+0.3) is not significant. For English-French, the subword regularization model obtained significantly worse results than our proposed model. The reason for this may be that both English and French are not morphologically complex. Combining our proposed approach with subword regularization improved the results non-significantly.

4.3 Vocabulary Size Experiments

When using BPE subword segmentation, the vocabulary size can be regulated. The results in Table 4 reveal the effects of varying vocabulary sizes.

The English-German NMT models are trained on approximately 4.5 million sentences and the English-Turkish models are trained on approximately 0.2 million sentences. Having a larger vocabulary leads to less word segmenting and shorter token sequences. When the training dataset is sufficiently large, a larger vocabulary results in higher BLEU scores. However, performance drops when there is insufficient data for training the rarest subwords. This effect is diminished by the proposed method because embeddings are inferred from more frequent sub-subword features.

The character-level models (char) yield poor BLEU scores, particularly for the low-resource settings. As shown by Cherry et al. (2018), character-level NMT requires deeper models.

Table 5 shows some statistics for English-Turkish models of different sizes. The encoder and decoder parameters of each model take 168.2 MiB. The size of the embedding matrix and the bias vector depends on the vocabulary size. The model size column shows the size of each model, baseline and proposed, after precomputing the embedding matrix and bias vector. The with FTE column shows the size of the proposed model without precomputing the embedding matrix and bias vector. This size depends on the number of features displayed in the column features.

| (full dataset) | char | 2K | 4K | 8K | 16K | 32K | 64K |
|----------------|------|----|----|----|-----|-----|-----|
| en → de        | baseline | 19.0 | 25.3 | 25.4 | 26.0 | 26.4 | 26.1 | 26.5 |
|                | proposed  | —   | 25.3 | 25.3 | 26.1 | 26.3 | 26.5 | 26.5 |
| en → tr        | baseline | 6.8 | 13.7 | 13.9 | **14.0** | 13.6 | 13.3 | 12.4 |
|                | proposed  | —   | 14.1 | 14.4* | 14.4* | 14.1 | 14.6* | **14.6** |
| tr → en        | baseline | 7.0 | 16.9 | 16.9 | **17.1** | 17.0 | 16.7 | 15.5 |
|                | proposed  | —   | 17.3 | 17.3 | 17.5 | 17.8* | 17.8* | **17.9** |

Table 4 Results for different vocabulary sizes. The English-German models are trained on a large corpus and the English-Turkish models are trained on a small corpus. The scores marked with * have a statistically significant improvement with respect to the best baseline.
Table 5  Statistics of English-Turkish models of different sizes. Model sizes are in MiB considering that each parameter takes 4 bytes. † The character-level model has a vocabulary size of 275 characters.

| vocabulary | features | model size | with FTE | test time | updates/sec |
|------------|----------|------------|----------|-----------|-------------|
| Char †     | —        | 169 MiB    | —        | 445.9 s   | 2.23        |
| BPE 2K     | 151      | 172 MiB    | 211 MiB  | 160.8 s   | 1.88 (97%)  |
| BPE 4K     | 194      | 176 MiB    | 214 MiB  | 137.8 s   | 1.86        |
| BPE 8K     | 234      | 184 MiB    | 218 MiB  | 122.0 s   | 1.82        |
| BPE 16K    | 301      | 200 MiB    | 230 MiB  | 102.9 s   | 1.74        |
| BPE 32K    | 367      | 231 MiB    | 257 MiB  | 101.0 s   | 1.59        |
| BPE 64K    | 470      | 293 MiB    | 325 MiB  | 101.4 s   | 1.37 (58%)  |

The test time column shows the seconds needed to translate the test set of 3,007 sentences. Larger vocabulary sizes produce shorter subword sequences, which results in faster decoding speeds. Decoding was done using beam search with a beam size of 5. The updates/sec column shows the times the parameters are updated per second during training. The penalty on training time for the proposed model is bigger for larger vocabulary sizes.

**Grammatical Mistakes**

Sennrich (2017) demonstrated that character-level NMT performs poorly in terms of morphosyntactic agreement as the distance between tokens increases. Their evaluation data, denoted as lingeval97, contain sentences with categorized synthetic errors. We evaluated our models to determine how vocabulary size affects grammatical accuracy and to elucidate the effects of the proposed method. The evaluate model is the large English-German corpus of approximately 4.5 M sentences. Table 6 lists accuracy metrics for different models for various error categories. Out of the 13 error categories, the proposed method performs the best on ten. The absolute best results are shown in bold font and the best results among the baseline models are underlined. The results are divided into four groups.

Some errors seem to be less common with smaller vocabulary sizes, even when the BLEU score is lower. When comparing each vocabulary size independently, we observe that our proposed model is superior to the baseline model with vocabulary sizes larger than 2 K subwords. The effect of including sub-subword information is larger for larger vocabulary sizes. As Table 4 shows, our proposed model does not significantly improve the BLEU scores for this large corpus. However, our proposed model is more resilient to the grammatical errors evaluated by lingeval97. The baseline and proposed models of 64 K subwords both have the same BLEU score of 26.5 but
the proposed model behaves better for 10 out of 13 grammatical mistake categories.

The first group of results is related to morpheme order (\textit{Compound}) and transliteration (\textit{Transl}). Small vocabularies work best for these error categories based on the nature of the target problem. Overall, including character n-gram information improves transliteration accuracy.

In the German language, the pronoun “Sie” can mean “they” or “she.” NMT models must choose the correct grammatical number for verbs by referring to source sentences. Including character n-gram features improves accuracy because the third-person singular English verbs in the source sentences include features indicating that they end with the letter “s.”

The third group of results is related to agreement. Morphosyntactic agreement between a subject and verb (\textit{SubjVerb}), and between determiners and nouns in noun phrases (\textit{NP}), considers number and gender. Agreement between verbs, particles (\textit{VerbPart}), and auxiliary verbs (\textit{Aux}) is improved by the proposed method. Excluding auxiliary verb agreement, these error categories are less common with longer sequences.

The final group of errors changes the polarity of a sentence by either inserting or deleting polarity markers. When the negating word “nicht” or other polarity affixes are inserted (\textit{NichtIns}, \textit{AffixIns}), the models operating on longer sequences perform better because they are more likely
to drop source sentence information.

There is no optimal vocabulary size. When considering only the BLEU scores, the larger vocabulary sizes are better. However this, improvements come as a trade-off with some error categories such as NichtIns and AffixIns.

4.4 Corpus Size Experiments

Table 7 compares the results of different methods for different dataset sizes. We sampled random sentences from the English-German training corpus to construct training sets of increasing sizes. We then trained NMT models using these sets. The vocabulary size was approximately 64,000 subwords. We trained three models for each dataset.

One can see that the benefit of applying the proposed method decreases for larger training datasets. For the full dataset of approximately 4.5 M sentences, the proposed method does not improve the results of the baseline model.

The models trained on smaller datasets are more prone to overfitting. We believe the proposed method provides better generalization because subword features are derived from n-gram features. This regularization effect is more prominent for small datasets. For large datasets, the need for regularization is lessened and the proposed method is less salient.

4.5 BT Experiments

BT (Sennrich et al. 2016b; Poncelas et al. 2018; Edunov et al. 2018) has been widely used to improve NMT. BT is particularly helpful for low-resource language pairs, but it requires a big monolingual corpus of the target language. A large monolingual corpus may not exist for some low-resource languages.

| sentences | baseline | proposed |
|-----------|----------|----------|
| small-1   | 265,483  | 17.1     | 19.2 (+2.1)* |
| small-2   | 410,966  | 19.9     | 21.4 (+1.5)* |
| small-3   | 753,437  | 22.5     | 23.1 (+0.6)* |
| small-4   | 1,506,874| 24.5     | 24.9 (+0.4)* |
| small-5   | 2,260,310| 25.4     | 25.7 (+0.3) |
| full      | 4,520,620| 26.5     | 26.5 (=)    |

Table 7  English-German NMT results for different dataset sizes. The benefit of the proposed method decreases as the size of the training dataset increases. The best results are shown in bold. It should be noted that the number of parameters for the proposed model after training is the same as that for the baseline. The scores marked with * have a statistically significant improvement with respect to their corresponding baseline.
Martinez et al. Sub-Subword N-Gram Features for Subword-Level Neural Machine Translation

| approach                        | BLEU  |
|---------------------------------|-------|
| best baseline (BPE 64K)         | 14.0  |
| best proposed (BPE 64K)         | 14.6  |
| BPE 32K baseline                | 13.3  |
| BPE 32K proposed                | 14.6  |
| baseline-baseline (BPE 32K)     | 17.3  |
| proposed-baseline (BPE 32K)     | 17.8* |
| proposed-proposed (BPE 32K)     | 17.7* |

Table 8 Results for different BT approaches for English-to-Turkish translation. Using the proposed approach to perform BT on the monolingual sentences to generate a synthetic corpus (proposed-baseline) yields the best performance. It should be noted that the number of parameters for the proposed model after training is the same as that for the baseline. The scores marked with * have a statistically significant improvement with respect to baseline-baseline.

We measured how the proposed approach behaves in combination with BT. Table 8 compares the BLEU scores for five English-to-Turkish NMT models. The first two rows contain the best results from Table 4. The three models using BT are represented by the remaining rows. These models have vocabulary sizes of approximately 32,000 words. We translated four million randomly sampled sentences from the Turkish monolingual common crawl corpus. The model baseline-baseline uses the baseline model with an 8,000 word vocabulary from Table 4 to perform BT on sentences. The models proposed-baseline and proposed-proposed use the proposed model with a vocabulary size of 64,000 words to perform BT on the monolingual sentences. We determined that there is no benefit to using the proposed method with synthetically enhanced training data.

5 Conclusion

We explored different approaches for utilizing character n-gram features for subword-level NMT without changing baseline model architectures or number of parameters. We analyzed the best approach for utilizing n-gram features and determined that a nonlinear function seems to perform better than a (weighted) sum of plural feature embeddings. For low-resource language pairs, translation accuracy (BLEU score) can be improved by utilizing n-gram features. This approach also improves the grammatical accuracy of the produced translations.

We explicitly analyzed how the vocabulary size affects grammatical accuracy and found that although models using larger vocabularies tend to produce fewer grammatical errors, they are difficult to train when training data is scarce. Using n-gram features to infer embeddings can help with training.
By varying the dataset size, we determined that the proposed method works particularly well on small datasets, but its benefits decrease as the dataset size increases. When combining the proposed method with BT, we determined that it is helpful to use the proposed method with a BT model, but it provides no benefits when applied to a forward translating model after incorporating a large synthetic dataset into the training data.

In the future, we wish to explore how the proposed approach can be applied to tasks other than NMT, such as named entity recognition. Additionally, it would be of interest to compare different feature selection algorithms.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). “Neural Machine Translation by Jointly Learning to Align and Translate.” *arXiv:1409.0473, [cs.CL]*.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). “Enriching Word Vectors with Subword Information.” *Transactions of the Association for Computational Linguistics, 5*, pp. 135–146.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A. (2014). “Findings of the 2014 Workshop on Statistical Machine Translation.” In *Proceedings of the 9th Workshop on Statistical Machine Translation*, pp. 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C., Negri, M., Post, M., Rubino, R., Specia, L., and Turchi, M. (2017). “Findings of the 2017 Conference on Machine Translation (WMT17).” In *Proceedings of the 2nd Conference on Machine Translation*, pp. 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Cherry, C., Foster, G., Bapna, A., Firat, O., and Macherey, W. (2018). “Revisiting Character-Based Neural Machine Translation with Capacity and Compression.” In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4295–4305.

Chung, J., Cho, K., and Bengio, Y. (2016). “A Character-Level Decoder without Explicit Segmentation for Neural Machine Translation.” *arXiv:1603.06147, [cs.CL]*.

Denkowski, M. and Neubig, G. (2017). “Stronger Baselines for Trustable Results in Neural Machine Translation.” In *Proceedings of the 1st Workshop on Neural Machine Translation*, pp. 18–27, Vancouver. Association for Computational Linguistics.
Martinez et al. Sub-Subword N-Gram Features for Subword-Level Neural Machine Translation

Edunov, S., Ott, M., Auli, M., and Grangier, D. (2018). “Understanding Back-Translation at Scale.” In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 489–500, Brussels, Belgium. Association for Computational Linguistics.

Hieber, F., Domhan, T., Denkowski, M., Vilar, D., Sokolov, A., Clifton, A., and Post, M. (2017). “Sockeye: A Toolkit for Neural Machine Translation.” arXiv:1712.05690, [cs.CL].

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). “Bag of Tricks for Efficient Text Classification.” In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 427–431, Valencia, Spain. Association for Computational Linguistics.

Koehn, P. (2004). “Statistical Significance Tests for Machine Translation Evaluation.” In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 388–395, Barcelona, Spain. Association for Computational Linguistics.

Koehn, P. (2005). “Europarl: A Parallel Corpus for Statistical Machine Translation.” In Conference Proceedings: The 10th Machine Translation Summit, pp. 79–86, Phuket, Thailand. AAMT, AAMT.

Kudo, T. (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, Vol. 1, pp. 66–75, Melbourne, Australia. Association for Computational Linguistics.

Lee, J., Cho, K., and Hofmann, T. (2016). “Fully Character-Level Neural Machine Translation without Explicit Segmentation.” arXiv:1610.03017, [cs.CL].

Morishita, M., Suzuki, J., and Nagata, M. (2018). “Improving Neural Machine Translation by Incorporating Hierarchical Subword Features.” In Proceedings of the 27th International Conference on Computational Linguistics, pp. 618–629.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). “Bleu: A Method for Automatic Evaluation of Machine Translation.” In Proceedings of 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Poncelas, A., Shterionov, D., Way, A., Wenniger, G. M. d. B., and Passban, P. (2018). “Investigating Backtranslation in Neural Machine Translation.” arXiv:1804.06189, [cs.CL].

Post, M. (2018). “A Call for Clarity in Reporting BLEU Scores.” arXiv:1804.08771, [cs.CL].

Press, O. and Wolf, L. (2017). “Using the Output Embedding to Improve Language Models.” In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 157–163, Valencia, Spain. Association for Computational Linguistics.
Computational Linguistics.

Schuster, M. and Nakajima, K. (2012). “Japanese and Korean voice search.” In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5149–5152.

Sennrich, R. (2017). “How Grammatical is Character-level Neural Machine Translation? Assessing MT Quality with Contrastive Translation Pairs.” In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 376–382, Valencia, Spain.

Sennrich, R., Haddow, B., and Birch, A. (2015). “Neural Machine Translation of Rare Words with Subword Units.” arXiv:1508.07909, [cs.CL].

Sennrich, R., Haddow, B., and Birch, A. (2016a). “Edinburgh Neural Machine Translation Systems for WMT 16.” In Proceedings of the 1st Conference on Machine Translation, pp. 371–376, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016b). “Improving Neural Machine Translation Models with Monolingual Data.” In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 86–96, Berlin, Germany. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). “Sequence to Sequence Learning with Neural Networks.” In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (Eds.), Advances in Neural Information Processing Systems 27, pp. 3104–3112. Curran Associates, Inc.

Takase, S., Suzuki, J., and Nagata, M. (2019). “Character N-gram Embeddings to Improve RNN Language Models.” In 33rd AAAI Conference on Artificial Intelligence.

Tiedemann, J. (2012). “Parallel Data, Tools and Interfaces in OPUS.” In Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC’12), Istanbul, Turkey. European Language Resources Association (ELRA).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. (2017). “Attention is All you Need.” In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30, pp. 5998–6008. Curran Associates, Inc.

Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2016). “Charagram: Embedding Words and Sentences via Character n-grams.” In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 1504–1515, Austin, Texas. Association for Computational Linguistics.

Wu, Y. and Zhao, H. (2018). “Finding Better Subword Segmentation for Neural Machine Trans-
Martinez et al. Sub-Subword N-Gram Features for Subword-Level Neural Machine Translation

Zhang, Z., Zhao, H., Ling, K., Li, J., Li, Z., He, S., and Fu, G. (2019). “Effective Subword Segmentation for Text Comprehension.” IEEE/ACM Transactions on Audio, Speech, and Language Processing, 27 (11), pp. 1664–1674.

Ander Martinez: From 2014 to 2019, he was a Research Assistant and Ph.D. candidate with the Division of Information Science of the Nara Institute of Science and Technology, Nara, Japan. He is currently a researcher at Fujitsu Laboratories, Kawasaki, Japan. His research interests include machine translation and dialogue modeling.

Katsuhito Sudoh: Associate professor at the Nara Institute of Science and Technology. He received his bachelor’s degree in engineering in 2000 and his master’s and Ph.D. degrees in informatics in 2002 and 2015, respectively, from Kyoto University. He worked with the NTT Communication Science Laboratories from 2002 to 2017.

Yuji Matsumoto: Team leader of the Knowledge Acquisition Team at Riken AIP. He received his M.S. degree and Ph.D. in information science from Kyoto University in 1979 and in 1990, respectively. He was a professor at Nara Institute of Science and Technology from 1993 to 2020.