SubMerge: Merging Equivalent Subword Tokenizations for Subword Regularized Models in Neural Machine Translation

Haiyue Song[1] **Francois Meyer**[2] **Raj Dabre**[1]
Hideki Tanaka[1] **Chenhui Chu**[3] **Sadao Kurohashi**[3,4]
[1] NICT, Japan [2] University of Cape Town, South Africa
[3] Kyoto University, Japan [4] NII, Japan
{haiyue.song, raj.dabre, hideki.tanaka}@nict.go.jp,
francois.meyer@uct.ac.za,
{chu, kuro}@i.kyoto-u.ac.jp

Abstract

Subword regularized models leverage multiple subword tokenizations of one target sentence during training. Previous decoding algorithms select one tokenization during inference, leading to the underutilization of knowledge learned about multiple tokenizations. To address this, we propose the **SubMerge** algorithm to rescue the ignored **Sub**word tokenizations through **Merg**ing equivalent ones during inference. SubMerge is a nested search algorithm where the outer beam search treats words as the minimal units, and the inner beam search provides a list of word candidates and their probabilities by merging subword tokenizations that form the same word. Experimental results on six machine translation datasets show more accurate word probability estimation and higher translation quality using SubMerge than beam search. Additionally, we provide time complexity analysis and investigate the effect of different beam sizes, training set sizes, dropout rates, and whether it is effective on non-regularized models.

1 Introduction

Despite the end-to-end nature that makes neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017; Gehring et al., 2017) the most prevalent and convenient approach for machine translation (MT), subword tokenization (Sennrich et al., 2016b; Provilkov et al., 2020; Kudo and Richardson, 2018; Kudo, 2018a) remains an indispensable preprocessing step for most NMT systems. Subword vocabularies address the out-of-vocabulary problem of word-based NMT systems (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2014; Luong et al., 2015) by reducing new words to known subwords, while avoiding the high computational cost of character-based NMT systems (Gupta et al., 2019; Kim et al., 2016; Costa-jussà and Fonollosa, 2016; Ling et al., 2015; Cherry et al., 2018) by enabling much shorter input and output sequences.

Deterministic segmenters like Byte-Pair Encoding (BPE) (Sennrich et al., 2016b) and Sentence-

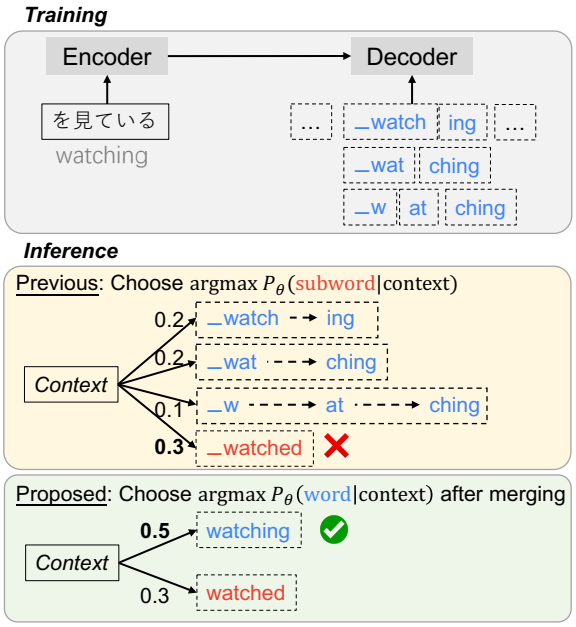

Figure 1: Subword regularized models suffer from discrepancies between training and inference, where they are trained on multiple target tokenizations and generate one. We propose to merge equivalent subword tokenizations that compose the same word with different conditional probabilities during the inference.

Piece (Kudo, 2018a) are widely used due to their simplicity and effectiveness. They are *determinisitic* in the sense that they consistently generate the same tokenization for a given sentence. NMT models trained on consistent subword tokenizations typically allocate the majority of a sentence's true probability (considering all potential tokenizations by marginalizing over them) to its specific tokenization (Cao and Rimell, 2021), except for out-of-domain data (Chirkova et al., 2023). Therefore, the probability of the sentence approximately equals the probability of that tokenization.

On the other hand, stochastic segmenters such as subword regularization methods (Provilkov et al., 2020; Kudo, 2018a) produce multiple tokenizations of a given sentence during training, as illustrated in Figure 1. As a data augmentation method, models trained on regularized data usually outperform those trained on non-regularized data, especially in low-resource scenarios. However, this causes a discrepancy between training and inference. During training, the model learns to generate multiple target tokenizations for each source sentence and learns to distribute the probability of a target sentence across all the tokenizations. During inference, greedy or beam search approximates the *single* highest probability tokenization. This causes a discrepancy - the probability of a target tokenization diverges drastically from the probability of a target sentence. The inaccurate probability estimation of the next word during inference in turn leads to a degradation in translation quality. The way to overcome this is to incorporate the marginal likelihood of the next words during decoding for the subword regularized models.

To this end, we propose SubMerge, a decoding algorithm that aggregates probabilities from exponentially many tokenizations for a sentence by merging subword tokenizations that form the same word. The property of BPE-dropout (Provilkov et al., 2020) that each word is individually segmented makes aggregating probabilities from exponentially many tokenizations theoretically possible. As for the implementation, SubMerge is a nested beam search approach. In the outer beam search, we hide the detail of possible subword tokenizations of the word, treating words as minimal units. This ensures that the outer beam is unaware of and unaffected by the subword tokenizer. In the inner beam search, we limit the search space within the word boundary. The inner beam search finds the n-best tokenizations, merges equivalent ones, and returns a list of words and the corresponding probabilities.

Previous attempts to estimate marginal likelihood over tokenizations include summing over n-best tokenizations (Cao and Rimell, 2021) and using importance sampling (Chirkova et al., 2023). However, these algorithms focus on perplexity estimation, assuming the output is already in hand. In our approach, we perform marginal likelihood estimation for the next words along with the inference process, aiming to improve not only the estimation precision but also the translation quality. In a nutshell, our contributions are as follows:

- We propose SubMerge, a nested beam search algorithm for generating text with subword regularized models. It merges equivalent subword tokenizations for the next words, thereby enhancing probability estimation precision and translation quality.
- Experimental results on six machine translation datasets demonstrate significant improvements in estimating the underlying word perplexity computation for a model and its translation quality.
- We provide analyses of time complexity, various beam sizes, the selection of the inner searching function, and the impact of hyperparameters.

2 Preliminaries

This section formulates the objective of the inference process of NMT models, highlights the distinction introduced by subword regularized models, and introduces how we address it.

Inference Objective An NMT model with parameters θ during inference is to obtain $\arg\max_Y P_\theta(Y|X)$ where X and Y are the source and target sentences in plain text form. For subword-based NMT models, we tokenize X into a sequence of tokens during both training and inference. We tokenize Y during the training and try to predict a sequence of tokens that compose Y during inference. We use two tokenizers $\tau_S(X) = \boldsymbol{x}$, where $\boldsymbol{x} = (x_1, ..., x_n)$ and $\tau_T(Y) = \boldsymbol{y}$, where $\boldsymbol{y} = (y_1, ..., y_m)$. Each subword x_i or y_i is a non-empty substring of the text X or Y in a finite-size subword vocabulary predefined by the source or

target tokenizer. In theory,

$$P_\theta(Y|X) \neq P_\theta(\boldsymbol{y}|\boldsymbol{x}), \quad (1)$$

because there are multiple tokenizations of X and Y (besides $\boldsymbol{x}$ and $\boldsymbol{y}$) that the model P_θ would assign non-zero probabilities to (Cao and Rimell, 2021).

Non-regularized Models For NMT models using deterministic tokenizers such as BPE (Sennrich et al., 2016b), tokenization function $\tau(\cdot)$ is a bijective function, and we can approximate the objective using one tokenization with a small gap (less than 0.5%) (Chirkova et al., 2023):

$$P_\theta(Y|X) \approx P_\theta(\boldsymbol{y}|\boldsymbol{x}). \quad (2)$$

Therefore, we can use $\arg\max_{\boldsymbol{y}} P_\theta(\boldsymbol{y}|\boldsymbol{x})$ to approximate $\arg\max_{Y} P_\theta(Y|X)$ with greedy or beam search in inference. This allows us to identify the next tokens with high conditional probabilities without concern for the discrepancy between the probability of raw text Y and of the particular tokenization $\boldsymbol{y}$.

Subword Regularized Models For NMT models using stochastic tokenizers (Provilkov et al., 2020), the tokenization function τ yields multiple tokenizations for one sentence. That is $\tau_S(X) = \boldsymbol{x} \in \mathcal{V}_\mathcal{S}(\mathcal{X})$ where $\boldsymbol{x} \sim P_{\tau_S}(\boldsymbol{x}|X)$. Similar for Y. In this case, the number of possible segmentation $\mathcal{V}_\mathcal{S}(\mathcal{X})$ increases exponentially according to the length of X, which deviates $P_\theta(\boldsymbol{y}|\boldsymbol{x})$ drastically from $P_\theta(Y|X)$, thus it requires marginalization over all possible tokenizations:

$$P_\theta(Y|X) = \sum_{\boldsymbol{x}\in\mathcal{V}_S(X)} \sum_{\boldsymbol{y}\in\mathcal{V}_T(Y)} P_\theta(\boldsymbol{y}|\boldsymbol{x}) P_{\tau_S}(\boldsymbol{x}|X). \quad (3)$$

This study focuses on better estimating the marginal likelihood of the target side, so we simplify Eq. (3) by using the most probable source tokenization $\arg\max_{\boldsymbol{x}\in\mathcal{V}_S(X)} P_{\tau_S(\cdot)}(\boldsymbol{x}|X)$ and remove the effect of the source tokenizer, resulting in:

$$P_\theta(Y|X) \approx \sum_{\boldsymbol{y}\in\mathcal{V}_T(Y)} P_\theta(\boldsymbol{y}|\boldsymbol{x}). \quad (4)$$

Inference for Subword Regularized Models We propose SubMerge to approximates Eq. (4) by introducing an intermediate variable, word tokenizations $\boldsymbol{w} = (w_1, ..., w_n)$, generated by a word tokenizer $\tau_W(\cdot)$ which is a bijective function.[1] The problem is simplified as:

$$P_\theta(Y|\boldsymbol{x}) = P_\theta(\boldsymbol{w}|\boldsymbol{x}) = \prod_{i=1}^{n} P_\theta(w_i|\boldsymbol{w}_{<i}, \boldsymbol{x}). \quad (5)$$

We estimate $P_\theta(w_i|\boldsymbol{w}_{<i}, \boldsymbol{x})$ by summing over probabilities of subword tokenizations for one word w_i where the search space is much smaller compared to the search space of tokenizations of a whole sentence in Eq. (4):

$$P_\theta(w_i|\boldsymbol{w}_{<i}, \boldsymbol{x}) \approx \sum_{\boldsymbol{y}'\in\mathcal{V}_T(w_i)} P_\theta(\boldsymbol{y}'|\boldsymbol{w}_{<i}, \boldsymbol{x}). \quad (6)$$

In practice, since the decoder only takes subword as input, we feed the best subword tokenization of the next word w_i, which is $\arg\max_{\boldsymbol{y}'\in\mathcal{V}_T(w_i)} P_\theta(\boldsymbol{y}'|\boldsymbol{w}_{<i})$. In this way, the probability of the target sentence is accurately calculated through a deterministic word tokenization as shown in Eq. (5), where the probability estimation of each word is precisely estimated through marginal likelihood estimation shown in Eq. (6). We implement Eq. (5) with the outer beam search as introduced in Section 3.2 and Eq. (6) with our inner beam search as introduced in Section 3.3.

3 Methodology

3.1 Overview of SubMerge

An overview of the SubMerge algorithm is shown in Figure 2. It is a nested beam search decoding algorithm that contains an outer beam search as explained in Section 3.2 and an inner beam search with subword merging post-processing as explained in Section 3.3. The outer beam search selects from a list of words considering the conditional probability in each step and estimates the most probable sentence $\arg\max_{Y} P_\theta(Y|X)$. The inner beam estimates the conditional probability of words in Eq. (6) by merging the probabilities of different subword tokenizations of the same words.

3.2 Outer Beam Search

The outer beam search algorithm is shown in Algorithm 1. It follows the standard beam search approach, where the difference is that words serve

[1] That is $\tau_W(Y) = \boldsymbol{w}$. Note that word tokenizer is not a bijective function for languages such as Japanese or Chinese. For these languages, we can use specific word segmentaters such as Jumanpp or Stanford Word Segmenter, which are bijective.

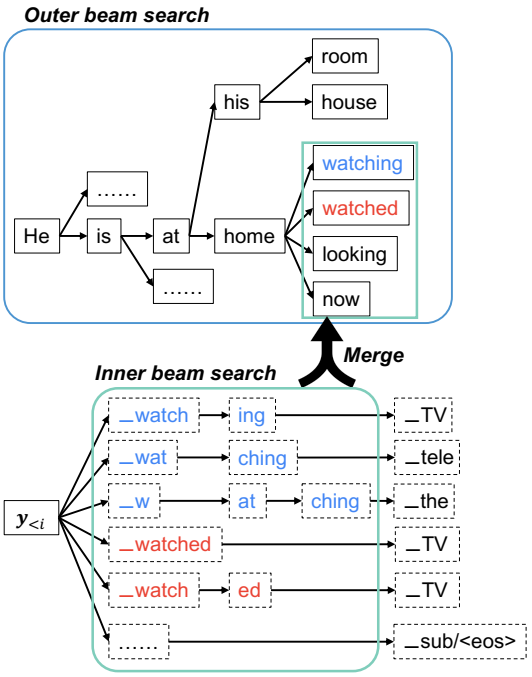

Figure 2: Overview of SubMerge. It contains an outer beam search that views words as minimal units. The candidate words and their probabilities are obtained from merging subword tokenizations in the n-best list of the inner beam search.

as the basic units. At each time step t in line 3, we explore each state s in the queue B_{t-1} that saves the best results in the previous step. When s is not finished, the candidates of the next words are obtained from a call to the inner beam search shown in line 9. Each state in the outer beam search queue contains the probability of the generation, the previous words, and their most probable tokens. Each state s' from the inner beam search contains the probability of the possible next word, the next word itself, and the most probable subword tokenization of that word. We add the new state to B_t shown in line 14 and only save the top-K ones shown in line 16. The most probable subword tokenization is used as the contextual input in the next decoding step.

In practice, we take the logarithm ($log(\cdot)$) of the probabilities for computational precision. We implemented early stopping after all sequences reach the special end-of-sentence ($< eos >$) token.

3.3 Inner Beam Search

The inner beam search is shown in Algorithm 2. It consists of two parts: 1) a token-level beam search within the word boundary and 2) post-processing to merge probabilities from equivalent subword tokenizations that compose the same word.

The first part is similar to that of the outer beam search. The stopping criteria of one sequence are reaching the start of the next word (with the start-of-word indicator '_' Unicode U+2581) or the $< eos >$ token, where this stopping token will not be added to the token list. Otherwise, the exploration of the sequence continues according to the next subword probability distribution given by the decoder. During the post-processing part, we remove special tokens and spaces during the detokenization of a token list to form the word and return a list of words with their probabilities. The time complexity of SubMerge is $O(T \cdot K^3)$, where T is the sentence length and K is the beam size with the derivation in Section 6.

Algorithm 1: OuterBeamSearch

Data: Beam width K, max length T
Result: Best sequence of states
1 **Initialization:**
2 $B_0 \leftarrow \{(0, [], [])\}$;
3 **for** $t \leftarrow 1$ **to** T **do**
4 $B_t \leftarrow \emptyset$;
5 **foreach** $s \in B_{t-1}$ **do**
6 **if** s *reaches* $< eos >$ **then**
7 $B_t.append(s)$;
8 continue;
9 **foreach** $s' \in$ *InnerBS*$(s[2])$ **do**
10 $score, word, toks = s'$;
11 $score \leftarrow s[0] + score$;
12 $words \leftarrow s[1] + words$;
13 $toks \leftarrow s[2] + toks$;
14 $B_t.append((score, words, toks))$;
15 Sort B_t by scores in descending order;
16 $B_t \leftarrow B_t[:K]$
17 **return** B_T

4 Experimental Setup

We introduce the MT datasets and pre-processing settings Section 4.1. In Section 4.2, we provide details around the model hyper-parameters, training and inference settings. In Section 4.3, we present our evaluation metrics.

4.1 Data and Pre-processing

Datasets We conducted MT experiments with datasets listed in Table 1, including WMT'22 Livonian–English (Liv–En), Asian Language Treebank (ALT), IWSLT'15 Vietnamese–English (Vi–En), WMT'16 Romanian–English (Ro–En), WMT'15 Finnish–English (Fi–En), and WMT'14 German–English (De–En) datasets. ALT is a multi-way parallel dataset containing data in English and other Asian languages including Filipino (Fil), Indonesian (Id), Japanese (Ja), Malay

Algorithm 2: InnerBeamSearch

Data: Beam width K, max length T, $toks$
Result: Next word list

Initialization:
$B_0 \leftarrow \{(0, toks)\}$;
for $t \leftarrow 1$ **to** T **do**
  $B_t \leftarrow \emptyset$;
  foreach $s \in B_{t-1}$ **do**
    if s *reaches* _ *or* $< eos >$ **then**
      $B_t.append(s)$;
      continue;
    foreach $s' \in Decoder(s[1])$ **do**
      $score, toks = s'$;
      $score \leftarrow s[0] + score$;
      $toks \leftarrow s[1] + toks$;
      $B_t.append((score, toks))$;
  Sort B_t by scores in descending order.;
  $B_t \leftarrow B_t[:K]$
$W = \{\}$;
foreach $s \in B_T$ **do**
  $score, toks = s$;
  $word = detokenize(toks)$;
  if $word \notin W$ **then**
    $W[word] = (score, toks)$
  else
    $W[word][0] += score$
return $list(W.items())$

Dataset	Train	Valid	Test
WMT'22 Liv–En	$1,127$	586	856
ALT Asian Langs–En	$18k$	$1,000$	$1,018$
IWSLT'15 Vi–En	$133k$	$1,553$	$1,268$
WMT'16 Ro–En	$612k$	$1,999$	$1,999$
WMT'15 Fi–En	$1.8M$	$1,500$	$1,370$
WMT'14 De–En	$4.5M$	$45,781$	$3,003$

Table 1: Statistics of the datasets.

(Ms), Vietnamese (Vi), and simplified Chinese (Zh). We used the ALT-standard-split tool[2] to split the dataset into train, validation, and test sets.

Data Pre-processing We performed word tokenization on all data. We applied Juman++ (Tolmachev et al., 2018) to data in Japanese, Stanford-tokenizer (Manning et al., 2014) to data in Chinese, and Moses tokenizer (Koehn et al., 2007) to data in other languages. We normalized Romanian data and removed diacritics following previous work (Sennrich et al., 2016a). We prepared the WMT'14 English–German dataset using a data cleaning and normalization tool from Fairseq.[3]

We applied subword tokenization to each translation direction separately. For source or target language, we trained a subword tokenizer with a subword vocabulary of $8k$ on the monolingual corpus from the training set. The vocabulary size is computed by the VOLT algorithm (Xu et al., 2021). For languages in WMT'22, ALT and IWSLT'15, they are $7k$ to $8k$, and for the remaining datasets they are $10k$ to $11k$. We used $8k$ for consistency. We applied a widely adopted toolkit[4] to train BPE-dropout tokenizers with a dropout rate of 0.2 for the generation of regularized data and train BPE tokenizers for the generation of non-regularized data. The dropout rate is selected through hyper-parameter grid search from 0.1 to 0.5 with steps of 0.1, where we found 0.2 usually optimal and rate ≥ 0.3 resulted in unstable training.

4.2 NMT Settings

Model We used the Fairseq framework (Ott et al., 2019). We refer model settings in previous works (Rubino et al., 2020; Provilkov et al., 2020). For WMT'22, ALT and IWSLT'15 datasets, we used 1 attention head, 6 decoder layers, and 4 or 6 encoder layers (4 layers only for En$\leftarrow$ $\rightarrow$Fil and Ja$\rightarrow$En) and FFN dim of 512. For other datasets we used the standard transformer base architecture (Vaswani et al., 2017). We set dropout and attention dropout rates to 0.1. We applied layer normalization (Lei Ba et al., 2016; Mao et al., 2023) for both the encoder and decoder.

Training We set the batch size to $3,072$ tokens for sentence in the source language and used eight GPUs, resulting in 25k tokens per batch. We used the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. We used warmup and linear decay for the learning rate (Vaswani et al., 2017), with 4k warm-up steps, an initial learning rate of $1.7 * 10^{-7}$ and a final learning rate of 5×10^{-4}. We used label smoothing for the cross entropy loss with $\epsilon_{ls} = 0.1$ (Szegedy et al., 2015). We calculated the loss on the validation set after each epoch and applied early stopping when no improvement was observed for 10 epochs.

SubMerge led to better word-level perplexities than traditional beam search and higher BLEU and chrF++ scores, often achieving statistically significant improvements.

[2] www2.nict.go.jp/astrec-att/member/mutiyama/ALT

[3] github.com/facebookresearch/fairseq/blob/main/examples/translation/prepare-wmt14en2de.sh

[4] github.com/google/sentencepiece

	Word Perplexity ↓		BLEU ↑		chrF++↑	
	Beam Search	SubMerge	Beam Search	SubMerge	Beam Search	SubMerge
Low-Resource Scenario						
WMT'22 Liv→En	5.93	**3.43**	1.52	**2.04**$_{+0.5}$	18.85	**19.45**$_{+0.6}$
WMT'22 En→Liv	19.39	**6.88**	2.70	**3.21**$_{+0.5}$	19.14	**19.41**$_{+0.3}$
ALT Fil→En	12.68	**4.59**	31.10	**31.82**$^{*}_{+0.7}$	57.98	**59.17**$_{+1.2}$
ALT En→Fil	9.56	**4.14**	30.20	**31.14**$^{*}_{+0.9}$	59.64	**60.14**$_{+0.5}$
ALT Id→En	17.91	**5.91**	27.35	**28.73**$^{*}_{+1.4}$	53.61	**56.39**$_{+2.8}$
ALT En→Id	16.44	**4.91**	33.63	**34.19** $_{+0.6}$	63.14	**63.89**$_{+0.8}$
ALT Ja→En	24.90	**7.79**	15.07	**15.26**$^{*}_{+0.2}$	45.07	**45.46**$_{+0.4}$
ALT En→Ja	6.55	**3.69**	14.38	**14.59** $_{+0.2}$	27.92	**29.02**$_{+1.1}$
ALT Ms→En	11.28	**4.33**	31.86	**32.16**$^{*}_{+0.3}$	59.01	**60.09**$_{+1.1}$
ALT En→Ms	12.82	**4.18**	38.83	**39.28** $_{+0.5}$	66.25	**66.91**$_{+0.7}$
ALT Vi→En	17.21	**6.14**	23.64	**24.97**$^{*}_{+1.3}$	52.32	**52.93**$_{+0.6}$
ALT En→Vi	8.64	**3.52**	27.35	**27.64** $_{+0.3}$	53.66	**53.82**$_{+0.2}$
ALT Zh→En	23.11	**7.81**	13.92	**14.31**$^{*}_{+0.4}$	43.54	**44.43**$_{+0.9}$
ALT En→Zh	13.61	**6.76**	9.03	**9.87**$^{*}_{+0.8}$	22.76	**23.25**$_{+0.5}$
Middle- and High- Resource Scenario						
IWSLT'15 Vi→En	14.41	**5.62**	27.87	**28.43**$^{*}_{+0.6}$	48.62	**50.59**$_{+2.0}$
IWSLT'15 En→Vi	7.98	**3.39**	28.08	**28.16** $_{+0.1}$	49.27	**50.18**$_{+0.9}$
WMT'16 Ro→En	7.44	**3.22**	**33.85**	33.77 $_{-0.1}$	58.75	**59.07**$_{+0.3}$
WMT'16 En→Ro	6.78	**3.11**	34.35	**34.50** $_{+0.1}$	58.66	**58.89**$_{+0.2}$
WMT'15 Fi→En	11.27	**4.27**	**18.95**	18.88 $_{-0.1}$	47.24	**47.55**$_{+0.3}$
WMT'15 En→Fi	22.52	**7.81**	16.51	**16.65** $_{+0.1}$	47.66	**47.97**$_{+0.3}$
WMT'14 De→En	10.33	**3.90**	28.85	**28.94** $_{+0.1}$	55.99	**56.52**$_{+0.5}$
WMT'14 En→De	12.74	**4.64**	24.69	**24.83** $_{+0.1}$	52.68	**52.77**$_{+0.1}$

Table 2: Results of Subword Regularized Models. Statistical significance $p < 0.01$ is indicated by * against Beam Search. SubMerge consistently improves over the Beam Search baseline in most directions. Word perplexity results represent the ability to accurately estimate sentence probability rather than fluency.

Inference We selected the checkpoint with the best loss on the validation set. We used beam search and SubMerge with a beam size of 4 without additional normalization techniques, such as length penalty or temperature sampling (Dong et al., 2022).

4.3 Evaluation Metrics

We report word perplexity on generated translations to compare the probabilities assigned to generations by models. To evaluate translation quality, we report BLEU using sacreBLEU (Post, 2018),[5] chrF++ (Popović, 2017),[6] and BLEURT (Appendix A). We performed paired bootstrap resampling for statistical significance tests (Koehn, 2004).

The word perplexity is calculated as follows. We first evaluate the negative log probability of the generated sentences for models using SubMerge by:

$$s_{score} = -\sum_{i} \log P_\theta(word_i), \tag{7}$$

and models with beam search by:

$$s_{score} = -\sum_{i} \log P_\theta(tok_i). \tag{8}$$

We evaluated the average word perplexity by

$$w_{ppl} = exp(\frac{1}{N} s_{score}), \tag{9}$$

where N is the number of words. We evaluated the word perplexity based on the generated hypothesis rather than the reference. This reflects the actual scenario in generation tasks where we dynamically generate the next token (word) conditioned on what the model has generated instead of on the ground truth. Nevertheless, word perplexity is a conditional probability dependent on not only the input but also the parameters in the model. Therefore, the perplexity results must always be considered along with model-independent metrics such as BLEU scores.

5 Translation Quality Results

The results for subword regularized models are shown in Table 2.

[5] BLEU+c.mixed+l.en-lang+#.1+s.exp+tok.13a+v.1.5.1

[6] `github.com/m-popovic/chrF` with c6w2F0.4. Similar trends were observed using different chrF settings.

Word Perplexity We observed that word perplexity results improved substantially in the regularized models in contrast to the tiny gap (0.5%) reported in the non-regularized models (Chirkova et al., 2023) and in our analysis shown in Section 7.5. This is due to the fact that multiple tokenizations for one word appeared during training, which acts as a label-smoothing function on multiple correct next tokens. Therefore, the probability weight is distributed across multiple subwords thus, it becomes necessary to incorporate the marginal likelihood. It is worth noting that here word perplexity represents the precision of probability estimation rather than fluency or quality of the output.

Translation Quality We also found translation quality improved, especially in low-resource scenarios where the average BLEU score improvement is 0.6, whereas in higher resource scenarios, it is 0.3. We also observed consistent improvement in the chrF++ score. While only one translation direction among higher resource directions is statistically significant, 8 out of 12 low-resource directions see statistically significant improvements. Furthermore, we observed that the improvement is greater for languages where words contain more subwords on average ($T_{subword}$). In the ALT dataset, each Japanese word contains an average of 1.59 subwords, resulting in a modest improvement of only 0.2 BLEU. In contrast, Filipino has a $T_{subword}$ of 2.16, leading to an improvement of 0.9 BLEU.

6 Efficiency

We show the theoretical analysis of time complexity as well as running time results of SubMerge comparing with beam search.

Time Complexity Let K denote beam size and T_{word} denote the number of words in the sentence. In the outer beam search Algorithm 1, the loop in line 3 contains at most T steps, and line 5 contains at most K steps. Therefore, the time complexity of the SubMerge is $O(T_{word} * K * O(InnerBeamSearch()))$.

In the inner beam search Algorithm 2, line 3 contains at most $T_{subword}$ steps, which is the number of subwords within the word boundary. Line 5 contains at most K steps, and line 9 contains at most K steps because each beam yields maximum K candidates by selecting top-K probable tokens. Therefore, the time complexity of Algorithm 2 is $O(T_{subword} * K * K)$.

The overall time complexity of SubMerge is $O(\sum_i (T_{subword}) * K^3) = O(T * K^3)$ which is K times slower than that of beam search which is $O(T * K^2)$.

Inference Time We compared the running times in the IWSLT'15 En→Vi direction using $K = 4$ extracted from the log data reported by the Fairseq framework. SubMerge took 1,665 seconds to generate 1,268 sentences, whereas beam search took 303 seconds, showing SubMerge is approximately 5.5 times slower. We set the batch size to 1 because the current SubMerge implementation does not yet support batch processing.

7 Analysis

We investigate the effect of different beam sizes on the algorithm in Section 7.1. Section 7.2 explores using a sampling algorithm as the inner search algorithm. Section 7.3 and Section 7.4 respectively analyze the impact of the training set size and the dropout rate. Section 7.5 show conditions in which SubMerge is effective.

7.1 Assessing Beam Sizes Variants

Figures 3 and 4 show the word perplexities and BLEU scores of using different beam sizes for both non-regularized models and subword regularized models, comparing beam search and SubMerge.

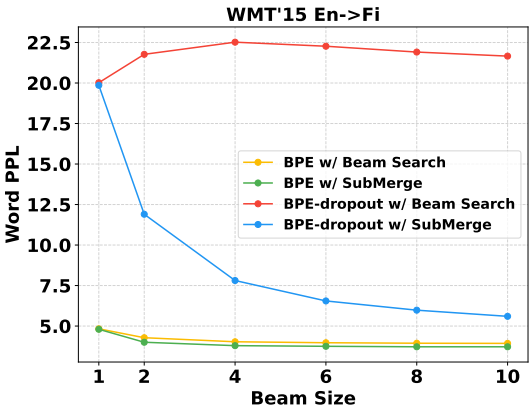

Figure 3: Word perplexity results using different beam sizes on the WMT'15 En→Fi direction.

We observed that as we increased the beam size, the word perplexity dropped sharply for BPE-dropout with SubMerge. When using a large beam size such as 10, it achieved comparable results to non-regularized models trained on one-best tokenization. Nevertheless, SubMerge does not yet accumulate as large a proportion of the

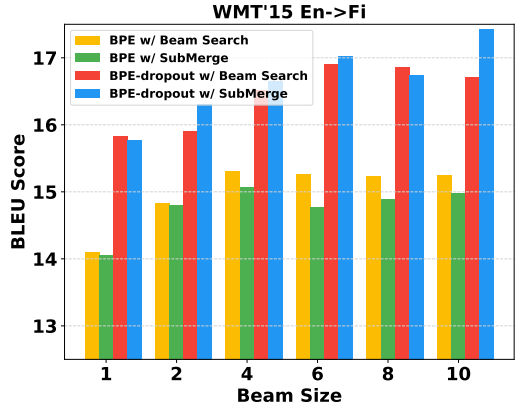

Figure 4: BLEU results using different beam sizes on the WMT'15 En→Fi direction.

probability distribution as using a non-regularized model. Since the training is on multiple segmentations, it certainly comes closer than when using beam search. For non-regularized models, combining equivalent paths for perplexity estimation also proved to be effective. We also observed that increasing beam size can lead to translation quality improvement for the SubMerge method. However, this is not the case for all directions (Cohen and Beck, 2019) and we put the full results using different beam sizes for all datasets in Appendix C.

7.2 Inner Search Algorithm Variants

We replaced the inner beam search with the sampling algorithm as shown in Algorithm 3. In the algorithm, Q is the queue that contains possible subword tokenizations of the next word. The sampling algorithm selects the next token tok_j in line 10 for each ongoing sample s according to the probability distribution of subwords in the target vocabulary outputted by the softmax function after the decoder. The current s is updated for both the score and the string. We call this pure sampling because we did not add sampling temperature, top-k or top-p filtering. We perform the merging post-processing the same as the inner beam search.

The word perplexity results are shown in Figure 5. For the sampling algorithm, we sampled n^2 tokenizations (where n is the beam size) in the inner loop and for each path, we started with the same historical information and selected the next subword according to the probability distribution until we reached the beginning of the next word. We then perform the same merging post-processing. However, we observed that the perplexity was higher than n-best tokenizations. This is because the sampling process could easily get lost at some step by selecting a token in the long tail with a very low probability.

Algorithm 3: InnerSampling

Data: Sample times K, max length T, $toks$
Result: Next word list
1 **Initialization:**
2 $s_0 \leftarrow \{(0, toks)\}$;
3 $Q \leftarrow \emptyset$;
4 **for** $i \leftarrow 1$ **to** K **do**
5 $s \leftarrow s_0$;
6 **for** $j \leftarrow 1$ **to** T **do**
7 **if** s *reaches _or* $< eos >$ **then**
8 $Q.append(s)$;
9 break;
10 Sample tok_j from $Decoder(s[1])$;
11 Update s using tok_j;
12 Sort Q by scores in descending order.;
13 $W = \{\}$;
14 **foreach** $s \in Q$ **do**
15 $score, toks = s$;
16 $word = detokenize(toks)$;
17 **if** $word \notin W$ **then**
18 $W[word] = (score, toks)$
19 **else**
20 $W[word][0] += score$
21 **return** $list(W.items())$

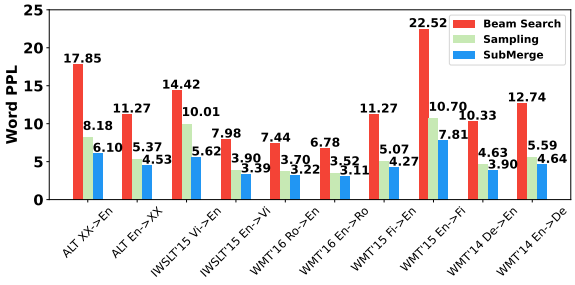

Figure 5: Word perplexity results comparing BPE-dropout with beam search to two variants of SubMerge: using either sampling as the inner search function or beam search.

7.3 Assessing Training Set Sizes

SubMerge is effective in extremely low-resource scenarios, as shown in Figure 6. We reported BLEU scores using beam search and SubMerge during decoding for models trained on 1k to 18k parallel sentences. SubMerge consistently outperformed beam search across training set sizes. Moreover, the BLEU improvement reached approximately 3.4 using only 1k data. This observation reveals the potential of SubMerge to be used in domain adaptation scenarios with limited data.

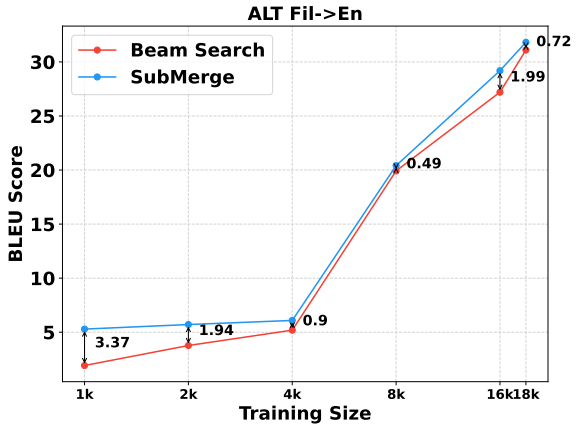

Figure 6: Translation quality using different sizes of training data. The x-axis is logarithmized.

7.4 Impact of Dropout Rates

Using a lower dropout rate in BPE-dropout yielded lower word perplexity and higher BLEU scores in higher resource scenarios, as shown in Table 3. When the dropout rate is low, the randomness of subword segmentation for a given word also decreases, leading to reduced variability in the training data and, concurrently, a diminished range of choices during the inference process. In the context of low-resource scenarios, reduced variability implies diminished data augmentation, which can adversely affect the model's generalization capability. Conversely, in higher resource settings, decreased variability signifies reduced noise, potentially enhancing model performance.

	Word PPL ↓		BLEU ↑	
Dropout Rate	0.1	0.2	0.1	0.2
ALT Others→En	**4.69**	6.10	22.06	**24.54**
ALT En→Others	**4.16**	4.53	24.75	**26.12**
IWSLT'15 Vi→En	**3.09**	5.62	**30.03**	28.43
IWSLT'15 En→Vi	**2.56**	3.39	**29.61**	28.16
WMT'16 Ro→En	**2.34**	3.22	**34.75**	33.77
WMT'16 En→Ro	**2.21**	3.11	**35.39**	34.50
WMT'15 Fi→En	**3.25**	4.27	18.87	**18.88**
WMT'15 En→Fi	**4.94**	7.81	16.64	**16.65**
WMT'14 De→En	**2.86**	3.90	**29.70**	28.94
WMT'14 En→De	**3.15**	4.64	**24.94**	24.83

Table 3: Results of SubMerge for models trained on BPE-dropout data with different dropout rates.

7.5 Does SubMerge Work on Non-regularized Models?

In short, No. We explored whether the proposed SubMerge method is applicable to non-regularized models using deterministic BPE tokenization. Table 4 presents word perplexities and BLEU scores on non-regularized models using beam search or SubMerge as the decoding algorithm.

	Word PPL ↓		BLEU ↑	
	BeamSearch	SubMerge	BeamSearch	SubMerge
WMT'22 Liv→En	3.60	**3.37**	0.36	**0.44**
WMT'22 En→Liv	5.22	**4.55**	0.64	**0.90**
ALT Others→En	6.02	**5.60**	**15.73**	15.40
ALT En→Others	4.90	**4.77**	**18.06**	17.82
IWSLT'15 Vi→En	2.95	**2.79**	24.34	**25.63**
IWSLT'15 En→Vi	2.43	**2.42**	**25.09**	24.86
WMT'16 Ro→En	2.14	**2.11**	**32.05**	31.70
WMT'16 En→Ro	2.00	**1.98**	**32.98**	32.85
WMT'15 Fi→En	2.85	**2.76**	**17.08**	16.94
WMT'15 En→Fi	4.03	**3.79**	**15.30**	15.06
WMT'14 De→En	**2.39**	2.40	**30.18**	30.04
WMT'14 En→De	2.45	**2.36**	**25.88**	25.71

Table 4: Results of non-regularized models trained on data using BPE tokenizer. We show the averaged results in En→ XX and XX→ En directions for the ALT dataset.

We observed lower word perplexity using SubMerge compared to using beam search. However, the improvement is not as significant (approximately 6%) as the improvement achieved by SubMerge for subword regularized models. This is consistent with our expectations. Models were trained on a single tokenization for each training word, so one tokenization accumulates the most probability weight. For the non-regularized model, results show the translation quality of SubMerge is not as good as that of beam search. Therefore, the proposed SubMerge method is only applicable to subword regularized models in the NMT task.

For other tasks, such as question answering, the word perplexity is greater because the task is less structured than MT, where the source sentence is a highly limiting constraint. For less constrained tasks, it is possible that SubMerge will improve the performance of even non-regularized models. We leave this for future work to explore.

8 Related Work

SubMerge is designed for decoding with text generation models for which likely tokenization probabilities diverge drastically from sentence probabilities. In other words, there are multiple tokenizations for one target sentence, and the probability distribution is splintered among them. Our objective is to enhance the inference algorithm on the target side. On the source side, merging probabilities of multiple tokenizations for a single source sentence has been shown to improve translation performance in low-resource scenar-

ios (Takase et al., 2022). Although we only experimented on models trained on data segmented by BPE-dropout (Provilkov et al., 2020), it also works for SentencePiece Regularization (Kudo, 2018a), MaxMatch-Dropout (Hiraoka, 2022) and NMT models with multiple subword segmenters (Kambhatla et al., 2022). On the other hand, NMT models trained on sentences segmented by deterministic segmenter only benefit from marginal likelihood estimation in out-of-domain data or long words (Cao and Rimell, 2021; Chirkova et al., 2023). Deterministic subword segmentation includes not only subword-level methods such as WordPiece (Schuster and Nakajima, 2012), BPE (Sennrich et al., 2016b), SentencePiece (Kudo and Richardson, 2018), dynamic programming encoding (He et al., 2020), BERT-Seg (Song et al., 2022), but also byte-level (Shaham and Levy, 2021), character-level (Tay et al., 2021), word-level (Mikolov et al., 2013), and hybrid word-character methods (Luong and Manning, 2016).

Marginal likelihood estimation can be implemented in two ways: sampling and dynamic programming. Sampling methods include summing over n-best tokenizations (Cao and Rimell, 2021) or important tokenizations (Chirkova et al., 2023). Sampling can be easily applied to any generation model. However, a manageable number of tokenizations cannot precisely estimate the probability of sentences with an exponentially large number of tokenizations, which is the case during the inference of the subword regularized models. On the other hand, dynamic programming can handle an exponentially large number of tokenizations by merging the same historical states, as introduced in *sequence modeling via segmentations* (Wang et al., 2017) and applied in the mixed-character-subword models (He et al., 2020; Meyer and Buys, 2023). However, they merge the historical states by approximating the previous output by character-level data. That is, after the decoder generates one subword, it is split into characters and fed to the decoder. This is not applicable to pure subword models. Based on the property that each word is individually segmented in BPE-dropout (Provilkov et al., 2020), we obtain n-best tokenizations within a small search space and treat the best tokenization of each word the historical state, taking advantage of both marginal likelihood estimation methods.

9 Conclusion and Future Work

We propose SubMerge to estimate the marginal likelihood of the next word by merging equivalent subword tokenizations during the inference of subword regularized models. Results demonstrate a significant improvement in word perplexity estimation and translation quality improvement in terms of BLEU and chrF++ scores, especially in low-resource scenarios.

Current inference algorithms are mostly based on conditional probability, which is a short-term value function. For future work of inference, we suggest aligning the value function towards evaluation metrics and human preference through reinforcement learning, where models are more aware of longer-term rewards.

Limitations

We did not experiment with common techniques in the beam search and SubMerge, such as length penalty. This is because we use a nested beam search, and the way to define the length (whether to use the number of tokens or the number of words) may differ from the definition in a traditional beam search. However, combining SubMerge with such techniques could be valuable for further work.

The word perplexity results reported in this paper are on the generated texts rather than reference texts. They do not correlate with fluency or translation quality, and we only use them to report how much of the probability weight of a model is being used during decoding, which is still useful.

We use the SentencePiece tool for the current implementation of BPE and BPE-dropout algorithms. Therefore, the SubMerge implementation is also based on the format of this specific tool, which uses "▁" (U+2581) to represent the beginning of a new word. However, other tools may use "@@" at the end of a subword to indicate that the current word has not ended yet. Therefore, the implementation of SubMerge may be slightly different in terms of ending conditions in the inner beam search.

We did not experiment on large-scale datasets (e.g., datasets with more than $100M$ parallel sentences). Reasons include 1) computational budget limitations and 2) the goal is verifying the algorithm rather than developing systems. We assume that the improvement will be marginal in high-resource scenarios.

References

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv e-prints*, page arXiv:1409.0473, September.

Cao, Kris and Laura Rimell. 2021. You should evaluate your language model on marginal likelihood over tokenisations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2104–2114, Online and Punta Cana, Dominican Republic, November. Association for Computational Linguistics.

Cherry, Colin, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, Brussels, Belgium, October-November. Association for Computational Linguistics.

Chirkova, Nadezhda, Germán Kruszewski, Jos Rozen, and Marc Dymetman. 2023. Should you marginalize over possible tokenizations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–12, Toronto, Canada, July. Association for Computational Linguistics.

Cohen, Eldan and Christopher Beck. 2019. Empirical analysis of beam search performance degradation in neural sequence models. In Chaudhuri, Kamalika and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1290–1299. PMLR, 09–15 Jun.

Costa-jussà, Marta R. and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany, August. Association for Computational Linguistics.

Dong, Chenhe, Yinghui Li, Haifan Gong, Miaoxin Chen, Junxin Li, Ying Shen, and Min Yang. 2022. A survey of natural language generation. *ACM Comput. Surv.*, 55(8), dec.

Gehring, Jonas, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1243–1252. JMLR.org.

Gupta, Rohit, Laurent Besacier, Marc Dymetman, and Matthias Gallé. 2019. Character-based nmt with transformer.

He, Xuanli, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic programming encoding for subword segmentation in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3042–3051, Online, July. Association for Computational Linguistics.

Hiraoka, Tatsuya. 2022. MaxMatch-dropout: Subword regularization for WordPiece. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4864–4872, Gyeongju, Republic of Korea, October. International Committee on Computational Linguistics.

Kalchbrenner, Nal and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October. Association for Computational Linguistics.

Kambhatla, Nishant, Logan Born, and Anoop Sarkar. 2022. Auxiliary subword segmentations as related languages for low resource multilingual translation. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, pages 131–140, Ghent, Belgium, June. European Association for Machine Translation.

Kim, Yoon, Yacine Jernite, David Sontag, and Alexander Rush. 2016. Character-aware neural language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar.

Kingma, Diederik P. and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Kudo, Taku and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November. Association for Computational Linguistics.

Kudo, Taku. 2018a. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July. Association for Computational Linguistics.

Kudo, Taku. 2018b. Subword regularization: Improving neural network translation models with multiple subword candidates.

Lei Ba, Jimmy, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv e-prints*, page arXiv:1607.06450, July.

Ling, Wang, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation.

Luong, Minh-Thang and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models.

Luong, Thang, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July. Association for Computational Linguistics.

Manning, Christopher, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.

Mao, Zhuoyuan, Raj Dabre, Qianying Liu, Haiyue Song, Chenhui Chu, and Sadao Kurohashi. 2023. Exploring the impact of layer normalization for zero-shot neural machine translation. In Rogers, Anna, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1300–1316, Toronto, Canada, July. Association for Computational Linguistics.

Meyer, Francois and Jan Buys. 2023. Subword segmental machine translation: Unifying segmentation and target sentence generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2795–2809, Toronto, Canada, July. Association for Computational Linguistics.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Ott, Myle, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Popović, Maja. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark, September. Association for Computational Linguistics.

Post, Matt. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October. Association for Computational Linguistics.

Provilkov, Ivan, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online, July. Association for Computational Linguistics.

Pu, Amy, Hyung Won Chung, Ankur P Parikh, Sebastian Gehrmann, and Thibault Sellam. 2021. Learning compact metrics for mt. In *Proceedings of EMNLP*.

Rubino, Raphael, Benjamin Marie, Raj Dabre, Atushi Fujita, Masao Utiyama, and Eiichiro Sumita. 2020. Extremely low-resource neural machine translation for asian languages. *Machine Translation*, 34(4):347–382.

Schuster, M. and K. Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany, August. Association for Computational Linguistics.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

Shaham, Uri and Omer Levy. 2021. Neural machine translation without embeddings. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, pages 181–186, Online, June. Association for Computational Linguistics.

Song, Haiyue, Raj Dabre, Zhuoyuan Mao, Chenhui Chu, and Sadao Kurohashi. 2022. BERTSeg: BERT based unsupervised subword segmentation for neural machine translation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 85–94, Online only, November. Association for Computational Linguistics.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Ghahramani, Z., M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision.

Takase, Sho, Tatsuya Hiraoka, and Naoaki Okazaki. 2022. Single model ensemble for subword regularized models in low-resource machine translation. In Muresan, Smaranda, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2536–2541, Dublin, Ireland, May. Association for Computational Linguistics.

Tay, Yi, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization.

Tolmachev, Arseny, Daisuke Kawahara, and Sadao Kurohashi. 2018. Juman++: A morphological analysis toolkit for scriptio continua. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 54–59, Brussels, Belgium, November. Association for Computational Linguistics.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Guyon, I., U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wang, Chong, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. 2017. Sequence modeling via segmentations.

Xu, Jingjing, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. 2021. Vocabulary learning via optimal transport for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7361–7373, Online, August. Association for Computational Linguistics.

A BLEURT Results

Table 5 shows BLEURT score results using the BLEURT-20 model (Pu et al., 2021). We can observe a similar trend with other metrics such as BLEU or chrF++, where the improvement is large in low-resource directions and comparable in higher-resource directions.

B Comparing with Non-Subword Models

We trained character-based and word-based models on IWSLT'15 Vi–En and WMT'16 Ro–En datasets and showed inferior performance compared to subword-based models using SubMerge as shown in Table 6. This conclusion is aligned with that in previous paper (Kudo, 2018b) and report.[7]

C Full Results for Different Beam Sizes

Tables 7, 8 and 9 show negative sentence log probability, word perplexity and BLEU scores for different beam sizes. The conclusions still remain the same where SubMerge improved probability estimation precision, which however did not bring translation quality improvement.

	BLEURT ↑	
	Beam Search	SubMerge
Low-Resource Scenario		
WMT'22 Liv→En	17.40	**17.47**
WMT'22 En→Liv	**42.74**	42.40
ALT Fil→En	55.35	**56.70**
ALT En→Fil	**47.95**	47.78
ALT Id→En	51.65	**53.91**
ALT En→Id	56.72	**57.10**
ALT Ja→En	41.54	**41.88**
ALT En→Ja	**27.02**	26.86
ALT Ms→En	56.87	**57.83**
ALT En→Ms	59.32	**59.44**
ALT Vi→En	49.61	**50.97**
ALT En→Vi	44.79	**45.11**
ALT Zh→En	40.95	**41.38**
ALT En→Zh	28.19	**29.19**
Middle- and High-Resource Scenario		
IWSLT'15 Vi→En	51.75	**52.57**
IWSLT'15 En→Vi	47.13	**47.46**
WMT'16 Ro→En	**61.52**	61.35
WMT'16 En→Ro	**52.08**	51.76
WMT'15 Fi→En	**57.12**	56.84
WMT'15 En→Fi	**53.67**	53.41
WMT'14 De→En	**60.01**	59.82
WMT'14 En→De	**54.97**	54.27

Table 5: BLEURT Results of Subword Regularized Models.

[7] github.com/google/sentencepiece/blob/master/doc/experiments.md

Models	Vi→En	En→Vi	Ro→En	En→Ro
Char-based	24.72	27.04	30.45	29.82
Word-based	21.40	25.24	26.33	25.67
Subword-based	28.43	28.16	33.77	34.50

Table 6: BLEU score results comparing different models on IWSLT'15 Vi–En and WMT'16 Ro–En datasets.

	IWSLT'15 Vi→En	IWSLT'15 En→Vi	WMT'16 Ro→En	WMT'16 En→Ro	WMT'15 Fi→En	WMT'15 En→Fi	WMT'14 De→En	WMT'14 En→De
BeamSize=1								
BPE w/ Beam Seach	25.67	24.37	19.86	19.08	23.64	23.30	21.91	21.01
BPE w/ SubMerge	24.82	24.06	22.87	20.62	23.40	23.07	28.98	20.25
BPE-dropout w/ Beam Seach	30.51	31.59	28.11	27.68	32.35	31.22	31.77	33.45
BPE-dropout w/ SubMerge	30.16	31.14	27.83	27.56	31.81	30.99	-	32.57
BeamSize=2								
BPE w/ Beam Seach	23.54	22.52	18.75	17.99	21.83	21.21	19.90	20.11
BPE w/ SubMerge	22.15	22.18	18.42	18.96	21.36	20.69	18.71	19.32
BPE-dropout w/ Beam Seach	29.74	30.58	27.93	27.22	31.84	30.55	31.25	33.03
BPE-dropout w/ SubMerge	25.83	27.15	23.95	23.45	27.52	26.75	26.59	28.72
BeamSize=3								
BPE w/ Beam Seach	22.95	21.90	18.39	17.53	21.18	20.43	19.31	19.75
BPE w/ SubMerge	21.33	21.66	18.12	18.17	20.66	19.95	18.44	18.97
BPE-dropout w/ Beam Seach	29.56	30.28	27.87	26.93	31.63	30.00	31.05	32.74
BPE-dropout w/ SubMerge	23.59	24.70	21.56	21.16	30.45	24.69	24.21	26.41
BeamSize=4								
BPE w/ Beam Seach	22.59	21.61	18.21	17.31	20.80	20.10	19.08	19.54
BPE w/ SubMerge	20.97	21.31	17.84	17.20	20.39	19.75	20.16	18.82
BPE-dropout w/ Beam Seach	29.52	30.06	27.79	26.79	31.46	29.71	30.88	32.57
BPE-dropout w/ SubMerge	22.57	23.65	20.35	19.86	23.82	23.72	22.91	25.08
BeamSize=5								
BPE w/ Beam Seach	22.43	21.42	18.07	17.18	20.50	19.81	18.88	19.43
BPE w/ SubMerge	20.66	21.14	17.70	17.64	20.16	19.53	19.33	18.70
BPE-dropout w/ Beam Seach	29.42	29.82	27.67	26.71	31.56	29.52	30.72	32.41
BPE-dropout w/ SubMerge	22.38	22.74	19.39	19.05	25.77	22.76	22.02	24.11
BeamSize=6								
BPE w/ Beam Seach	22.21	21.31	18.01	17.06	20.42	19.67	18.77	19.37
BPE w/ SubMerge	20.46	20.96	17.69	16.97	20.09	19.41	19.18	18.62
BPE-dropout w/ Beam Seach	29.42	29.65	27.67	26.65	31.20	29.41	30.66	32.36
BPE-dropout w/ SubMerge	21.75	22.31	18.82	18.38	22.61	22.23	21.41	23.49
BeamSize=8								
BPE w/ Beam Seach	21.95	21.13	17.85	16.87	20.21	19.45	18.66	19.25
BPE w/ SubMerge	20.26	20.77	17.53	17.44	19.80	19.18	18.74	18.51
BPE-dropout w/ Beam Seach	29.08	29.51	27.57	26.57	31.19	29.18	30.49	32.14
BPE-dropout w/ SubMerge	21.20	22.07	18.28	17.87	21.80	21.65	20.83	22.80
BeamSize=10								
BPE w/ Beam Seach	21.77	20.93	17.75	16.72	20.06	19.24	18.60	19.16
BPE w/ SubMerge	20.08	20.62	17.41	17.06	19.69	19.02	18.41	18.39
BPE-dropout w/ Beam Seach	28.84	29.39	27.46	26.47	30.69	29.02	30.40	32.03
BPE-dropout w/ SubMerge	20.82	21.56	17.84	17.44	22.03	21.12	20.37	22.26

Table 7: Negative sentence log probability of the generated hypothesis using different beam sizes.

	IWSLT'15 Vi→En	IWSLT'15 En→Vi	WMT'16 Ro→En	WMT'16 En→Ro	WMT'15 Fi→En	WMT'15 En→Fi	WMT'14 De→En	WMT'14 En→De
BeamSize=1								
BPE w/ Beam Seach	3.34	2.67	2.27	2.12	3.19	4.83	2.64	2.58
BPE w/ SubMerge	3.30	2.67	2.55	2.26	3.20	4.80	3.20	2.52
BPE-dropout w/ Beam Seach	4.42	3.42	3.20	2.99	4.79	8.32	4.19	4.59
BPE-dropout w/ SubMerge	4.42	3.41	3.18	3.00	4.76	8.28	-	4.47
BeamSize=2								
BPE w/ Beam Seach	3.05	2.50	2.18	2.04	2.95	4.28	2.45	2.50
BPE w/ SubMerge	2.92	2.48	2.15	2.10	2.88	4.00	2.34	2.39
BPE-dropout w/ Beam Seach	4.30	3.29	3.19	2.95	4.76	8.01	4.13	4.54
BPE-dropout w/ SubMerge	3.56	2.92	2.70	2.53	3.85	6.06	3.35	3.68
BeamSize=3								
BPE w/ Beam Seach	2.98	2.45	2.15	2.01	2.88	4.09	2.41	2.47
BPE w/ SubMerge	2.84	2.44	2.12	2.05	2.79	3.83	2.30	2.36
BPE-dropout w/ Beam Seach	4.31	3.26	3.19	2.92	4.77	7.82	4.12	4.51
BPE-dropout w/ SubMerge	3.25	2.67	2.46	2.32	4.29	5.28	3.02	3.34
BeamSize=4								
BPE w/ Beam Seach	2.95	2.43	2.14	2.00	2.85	4.03	2.39	2.45
BPE w/ SubMerge	2.79	2.42	2.11	1.98	2.76	3.79	2.40	2.36
BPE-dropout w/ Beam Seach	4.31	3.26	3.19	2.91	4.77	7.74	4.11	4.50
BPE-dropout w/ SubMerge	3.09	2.56	2.34	2.21	3.25	4.94	2.86	3.15
BeamSize=5								
BPE w/ Beam Seach	2.93	2.42	2.13	1.99	2.83	3.98	2.38	2.45
BPE w/ SubMerge	2.77	2.41	2.10	2.02	2.75	3.75	2.35	2.35
BPE-dropout w/ Beam Seach	4.26	3.24	3.18	2.91	4.80	7.68	4.10	4.49
BPE-dropout w/ SubMerge	2.97	2.52	2.26	2.15	3.48	4.68	2.75	3.03
BeamSize=6								
BPE w/ Beam Seach	2.92	2.41	2.13	1.99	2.82	3.97	2.38	2.44
BPE w/ SubMerge	2.76	2.40	2.10	1.97	2.75	3.75	2.35	2.35
BPE-dropout w/ Beam Seach	4.27	3.23	3.18	2.91	4.76	7.64	4.10	4.49
BPE-dropout w/ SubMerge	2.88	2.46	2.21	2.09	3.09	4.51	2.68	2.95
BeamSize=8								
BPE w/ Beam Seach	2.90	2.40	2.12	2.00	2.82	3.94	2.38	2.44
BPE w/ SubMerge	2.74	2.39	2.09	2.01	2.74	3.72	2.32	2.34
BPE-dropout w/ Beam Seach	4.23	3.22	3.19	2.90	4.76	7.56	4.08	4.48
BPE-dropout w/ SubMerge	2.82	2.43	2.16	2.05	2.98	4.35	2.63	2.87
BeamSize=10								
BPE w/ Beam Seach	2.89	2.40	2.12	2.00	2.82	3.93	2.38	2.44
BPE w/ SubMerge	2.74	2.39	2.09	1.98	2.73	3.72	2.32	2.34
BPE-dropout w/ Beam Seach	4.20	3.22	3.17	2.89	4.69	7.51	4.08	4.46
BPE-dropout w/ SubMerge	2.79	2.41	2.13	2.03	3.03	4.22	2.58	2.82

Table 8: Word perplexity of the generated hypothesis using different beam sizes.

	IWSLT'15 Vi→En	IWSLT'15 En→Vi	WMT'16 Ro→En	WMT'16 En→Ro	WMT'15 Fi→En	WMT'15 En→Fi	WMT'14 De→En	WMT'14 En→De
BeamSize=1								
BPE w/ Beam Seach	23.68	24.44	31.34	32.53	16.61	14.10	29.34	25.16
BPE w/ SubMerge	24.22	24.18	31.08	32.14	16.55	14.05	26.45	24.91
BPE-dropout w/ Beam Seach	29.28	28.52	34.45	34.87	18.21	16.09	29.08	24.46
BPE-dropout w/ SubMerge	29.13	28.56	34.10	34.53	18.31	16.28	-	24.31
BeamSize=2								
BPE w/ Beam Seach	23.98	24.92	31.82	32.79	17.06	14.83	30.05	25.63
BPE w/ SubMerge	25.40	24.80	31.45	32.62	17.10	14.80	30.19	25.59
BPE-dropout w/ Beam Seach	29.87	29.21	35.02	35.25	18.64	16.70	29.48	24.73
BPE-dropout w/ SubMerge	30.01	29.16	34.56	35.19	18.80	16.30	29.53	24.72
BeamSize=3								
BPE w/ Beam Seach	24.44	24.92	32.03	33.02	16.89	15.30	30.25	25.84
BPE w/ SubMerge	25.47	24.94	31.66	33.00	16.97	14.84	30.23	25.75
BPE-dropout w/ Beam Seach	29.77	29.34	35.06	35.55	18.77	16.98	29.58	24.89
BPE-dropout w/ SubMerge	29.64	29.20	34.45	35.36	18.26	16.49	29.51	24.84
BeamSize=4								
BPE w/ Beam Seach	24.34	25.09	32.05	32.98	17.08	15.30	30.18	25.88
BPE w/ SubMerge	25.63	24.86	31.70	32.85	16.94	15.06	30.04	25.71
BPE-dropout w/ Beam Seach	29.65	29.40	34.96	35.44	18.80	16.95	29.75	24.79
BPE-dropout w/ SubMerge	30.03	29.61	34.75	35.39	18.87	16.64	29.70	24.94
BeamSize=5								
BPE w/ Beam Seach	24.38	25.02	32.02	32.95	17.05	15.26	30.13	25.80
BPE w/ SubMerge	25.79	24.93	31.75	33.00	17.07	14.89	30.08	25.67
BPE-dropout w/ Beam Seach	29.36	29.44	34.99	35.55	18.97	17.14	29.69	24.82
BPE-dropout w/ SubMerge	29.50	28.67	34.48	35.43	18.37	16.74	29.61	24.87
BeamSize=6								
BPE w/ Beam Seach	24.45	25.07	31.99	32.88	17.02	15.28	30.11	25.78
BPE w/ SubMerge	25.58	24.86	31.62	32.85	17.04	14.77	30.07	25.67
BPE-dropout w/ Beam Seach	29.35	29.53	34.96	35.47	19.05	17.26	29.61	24.81
BPE-dropout w/ SubMerge	29.58	29.18	34.46	35.32	14.41	16.99	29.50	24.91
BeamSize=8								
BPE w/ Beam Seach	24.57	24.81	31.86	32.37	17.14	15.23	30.06	25.68
BPE w/ SubMerge	25.86	24.83	31.72	32.81	16.78	14.89	29.93	25.63
BPE-dropout w/ Beam Seach	29.33	29.52	34.99	35.58	18.95	17.21	29.54	24.73
BPE-dropout w/ SubMerge	29.73	29.52	34.76	35.27	18.88	17.06	29.46	24.87
BeamSize=10								
BPE w/ Beam Seach	24.77	24.93	31.78	32.05	17.09	15.24	30.09	25.75
BPE w/ SubMerge	25.93	24.66	31.63	32.85	16.69	14.97	29.90	28.16
BPE-dropout w/ Beam Seach	29.32	29.41	34.97	35.59	19.08	17.32	29.58	24.80
BPE-dropout w/ SubMerge	29.39	28.92	34.45	35.39	18.88	16.92	29.28	24.68

Table 9: BLEU scores on test sets using different beam sizes.