BERTSeg: BERT Based Unsupervised Subword Segmentation for Neural Machine Translation

Haiyue Song[1,2] **Raj Dabre**[2] **Zhuoyuan Mao**[1]
Chenhui Chu[1] **Sadao Kurohashi**[1]
[1] Kyoto University, Japan [2] NICT, Japan
{song, zhuoyuanmao, chu, kuro}@nlp.ist.i.kyoto-u.ac.jp
raj.dabre@nict.go.jp

Abstract

Existing subword segmenters are either 1) frequency-based without semantics information or 2) neural-based but trained on parallel corpora. To address this, we present **BERTSeg**, an unsupervised neural subword segmenter for neural machine translation, which utilizes the contextualized semantic embeddings of words from characterBERT and maximizes the generation probability of subword segmentations. Furthermore, we propose a generation probability-based regularization method that enables BERTSeg to produce multiple segmentations for one word to improve the robustness of neural machine translation. Experimental results show that BERTSeg with regularization achieves up to 8 BLEU points improvement in 9 translation directions on ALT, IWSLT15 Vi→En, WMT16 Ro→En, and WMT15 Fi→En datasets compared with BPE. In addition, BERTSeg is efficient, needing up to 5 minutes for training.

BERTSeg Segmentation	
watch/ing	un/break/able
leak/ed	wave/length/s
stress/ful	share/holding/s
employ/er/s	ab/normal/ly

Table 1: BERTSeg produces linguistically intuitive subword semgnetations.

BERTSeg-Regularization Segmentation	
represent/ed	represented
represent/e/d	re/presented
re/presented	re/present/e/d

Table 2: BERTSeg-Regularization samples multiple segmentations from one word.

1 Introduction

Subword segmentation is the task of splitting a word into smaller n-gram character units called subwords (Schuster and Nakajima, 2012). It alleviates the out-of-vocabulary (OOV) problem in neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017) by enabling an NMT system to have a fixed-size vocabulary while being able to handle all possible words regardless of their frequencies.

Studies in subword segmentation fall into two categories: frequency-based approaches and neural network-based approaches. Frequency-based approaches (Sennrich et al., 2016; Kudo and Richardson, 2018; Kudo, 2018; Provilkov et al., 2020) adopt a greedy algorithm that generates the vocabulary with frequent subword fragments in the corpus during training and merges adjacent high-frequency fragments starting from characters recursively during inference. Among these methods, BPE-dropout (Provilkov et al., 2020) and SentencePiece with regularization (Kudo, 2018) generate multiple segmentations by random sampling. Frequency-based approaches do not consider semantic information of the subwords, therefore the generated segmentation is not linguistically motivated. For example, the word “fellowships” is segmented into “fell/ows/hip/s” by BPE whereas “fellow/ships” is a more linguistically motivated segmentation. Neural approaches such as DPE (He et al., 2020) implicitly considers the contextual semantic information of subwords by maximizing the generation probabilities of the target language sentences conditioned on the source language sentences. However, it trains on parallel sentences, which poses a problem for low-resource languages. DPE is slow because it calculates the probabilities of all possible sentence segmentations, therefore, not practical in high-resource scenarios.

We propose BERTSeg, an unsupervised neural subword segmenter that leverages contextualized word representations from the pre-trained model, characterBERT (El Boukkouri et al., 2020). It combines the advantages of frequency-based and neural

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
November 20–23, 2022. ©2022 Association for Computational Linguistics

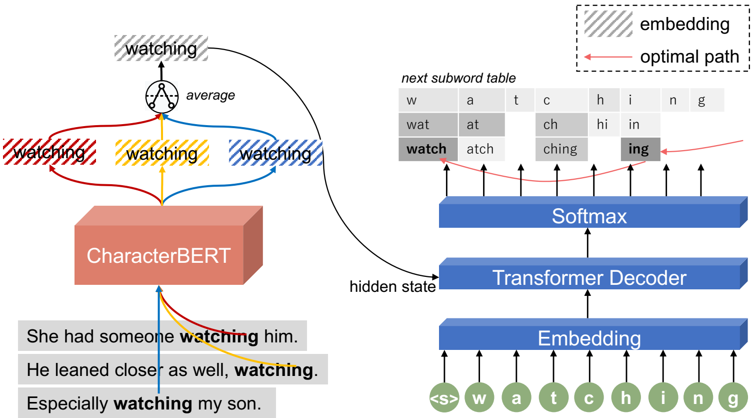


Figure 1: **BERTSeg architecture.** The encoder is a characterBERT that generates average embeddings for one word in different contexts. The transformer decoder takes characters as input and generates probabilities of the next subword. During training, the objective is to maximize the probabilities of all possible segmentations. During inference, the model retraces the optimal segmentation.

approaches by 1) leveraging word-level monolingual data and 2) capturing semantic information explicitly. The semantic information is provided by characterBERT, which has been shown to be helpful for natural language understanding tasks. In our task, this enables the model to generate linguistically intuitive segmentations rather than high-frequency fragments, as shown in Table 1.

Furthermore, we propose a subword regularization method BERTSeg-Regularization which enables the model to produce multiple segmentations based on segmentation probabilities to improve the robustness of NMT, as represented in Table 2.

Experimental results on the low-resource ALT and high-resource IWSLT and WMT datasets show approximately 5 and 2 BLEU points improvement over BPE with statistical significance $p < 0.001$ and outperforms all other baseline methods. Moreover, our method is efficient because of leveraging the word-level data. BERTSeg requires up to 5 minutes to train, whereas DPE requires hours to days to train and VOLT also costs 30 minutes to generate the optimal vocabulary. Finally, analysis shows high generalizability on unseen words.

2 Methodology

2.1 Background: Word Modeling

We define a word as a single distinct meaningful element of writing. Technically, we split words in sentences with tools for different languages as described in Section 3. Let $\boldsymbol{x}_{1:T}$ denote a word containing T characters. $\boldsymbol{a}_{1:\tau_a}$ is one segmentation of $\boldsymbol{x}$ that comprises τ_a subwords a_i. $\mathcal{S}(\boldsymbol{x})$ is the set of all possible segmentations of $\boldsymbol{x}$. The generation probability $\boldsymbol{x}$ can be defined as the sum of the probabilities of all segmentations shown in Eq. (1).

$$\begin{aligned} p(\boldsymbol{x}_{1:T}) &= \sum_{\boldsymbol{a}_{1:\tau_a} \in \mathcal{S}(\boldsymbol{x})} p(\boldsymbol{a}_{1:\tau_a}) \\ &= \sum_{\boldsymbol{a}_{1:\tau_a} \in \mathcal{S}(\boldsymbol{x})} \prod_{i=1}^{\tau_a} p(a_i | a_1, \ldots, a_{i-1}) \end{aligned} \tag{1}$$

2.2 Proposed Method: BERTSeg

As shown in Figure 1, the proposed BERTSeg contains a characterBERT encoder (El Boukkouri et al., 2020) and a mixed character-subword transformer decoder (He et al., 2020). The mixed character-subword transformer takes characters as input and generates sub-words as output. The model represents the history information by prefix characters $x_1, \ldots, x_j$ instead of previous subwords $a_1, \ldots, a_{i-1}$, where j is the index of the last character in a_{i-1}.

Let $\boldsymbol{e}_{\boldsymbol{x}}$ denote the average-pooled contextualized word embeddings by characterBERT from all sentences containing word $\boldsymbol{x}$. The generation probability can be calculated by Eq. (2).

$$\begin{aligned} &\log p(\boldsymbol{x}_{1:T} | \boldsymbol{e}_{\boldsymbol{x}}) = \\ &\log \sum_{\boldsymbol{a}_{1:\tau_a} \in \mathcal{S}(\boldsymbol{x})} \prod_{i=1}^{\tau_a} p(a_i | \boldsymbol{e}_{\boldsymbol{x}}; x_1, \ldots, x_j) \end{aligned} \tag{2}$$

During training, we calculate the $\log p(\boldsymbol{x}_{1:T} | \boldsymbol{e}_x)$ in polynomial time by dynamic programming (DP) (He et al., 2020) and use $-\log p(\boldsymbol{x}_{1:T} | \boldsymbol{e}_x)$ as the loss. During inference, we retrace the optimal segmentation $\boldsymbol{a}$ through Eq. (3).

$$\boldsymbol{a} = \arg\max_{\boldsymbol{a}_{1:\tau_a} \in \mathcal{S}(\boldsymbol{x})} \prod_{i=1}^{\tau_a} p(a_i | \boldsymbol{e}_x; x_1, \ldots, x_j) \tag{3}$$

	Fil→En	Id→En	Ja→En	Ms→En	Vi→En	Zh→En	Avg
w/o Regularization							
BPE (Sennrich et al., 2016)	23.09	25.70	9.42	28.19	19.94	12.21	19.76
VOLT (Xu et al., 2021)	22.99	25.05	10.56	27.91	21.64	11.31	19.91
DPE (He et al., 2020)	24.04	26.66	9.93	27.89	20.06	10.72	19.88
BERTSeg	$24.84^{*}_{+1.8}$	$25.84_{+0.1}$	$10.97^{*\circ}_{+1.6}$	$29.52^{*\circ}_{+1.3}$	$20.86_{+0.9}$	$12.20^{\circ}_{-0.0}$	$20.71_{+1.0}$
With Regularization							
BPE-dropout (Provilkov et al., 2020)	28.18	28.02	12.84	31.59	23.67	13.91	23.04
BERTSeg-Regularization	$\mathbf{31.09}^{*\circ}_{+8.0}$	$\mathbf{28.86}^{*\circ}_{+3.2}$	$\mathbf{15.56}^{*\circ}_{+6.1}$	$\mathbf{32.97}^{*\circ}_{+4.8}$	$\mathbf{24.58}^{*\circ}_{+4.6}$	$\mathbf{15.03}^{*\circ}_{+2.8}$	$\mathbf{24.68}_{+4.9}$

Table 3: **Low-resource Asian languages→English MT BLEU score results.** BERTSeg-Regularization consistently improves over all baselines. Statistical significance $p < 0.001$ is indicated by * against BPE and by $^\circ$ against DPE. Subscript values denote the BLEU score differences from BPE.

	Fil→En	Id→En	Ja→En	Ms→En	Vi→En	Zh→En	Avg
w/o Regularization							
BPE (Sennrich et al., 2016)	29.05	31.05	20.12	32.74	27.64	22.85	27.24
VOLT (Xu et al., 2021)	29.16	30.98	21.24	32.50	28.37	22.22	27.41
DPE (He et al., 2020)	29.72	31.79	21.13	32.50	26.94	21.46	27.26
BERTSeg	$30.28_{+1.2}$	$31.25_{+0.2}$	$21.04_{+0.9}$	$33.34_{+0.6}$	$27.38_{-0.3}$	$22.57_{-0.3}$	$27.64_{+0.4}$
With Regularization							
BPE-dropout (Provilkov et al., 2020)	31.96	32.99	22.83	34.81	29.05	23.56	29.20
BERTSeg-Regularization	$\mathbf{34.35}_{+5.3}$	$\mathbf{33.38}_{+2.3}$	$\mathbf{25.14}_{+5.0}$	$\mathbf{36.13}_{+3.4}$	$\mathbf{30.40}_{+2.8}$	$\mathbf{24.57}_{+1.7}$	$\mathbf{30.66}_{+3.4}$

Table 4: **Low-resource Asian languages→English MT METEOR score results.** BERTSeg-Regularization consistently improves over all baselines. Subscript values denote the BLEU score differences from BPE.

2.3 Probability Based Regularization

We propose BERTSeg-Regularization which performs subword regularization based on the probability distribution during inference. For segmentation $\boldsymbol{a}_i$ with $p(\boldsymbol{a}_i)$, the sampling probability $p_{sample}(\boldsymbol{a}_i)$ is shown in Eq. (4), where t is a temperature hyperparameter.

$$p_{sample}(\boldsymbol{a}_i) = \frac{e^{\log p(\boldsymbol{a}_i)/t}}{\sum_{\boldsymbol{a}_i \in \mathcal{S}(\boldsymbol{x})} e^{\log p(\boldsymbol{a}_i)/t}} \tag{4}$$

The time complexity for generating the best N segmentations is $O(N \log NT^2)$ through DP.

3 Experimental Settings

Datasets Table 5 summarizes MT datasets from low- to high-resource. We use the English words of each dataset to train BERTSeg. We applied Juman++ (Tolmachev et al., 2018) to Japanese sentences, Stanford-segmenter (Manning et al., 2014) to Chinese sentences, and Moses tokenizer (Koehn et al., 2007) to sentences in other languages. We removed diacritics in Romanian sentences. We set the subword vocabulary size to $8k$ for all segmentation methods and NMT models.

Dataset	Train	Valid	Test
ALT Asian Langs-En	$18k$	$1,000$	$1,018$
IWSLT15 Vi-En	$133k$	$1,553$	$1,268$
WMT16 Ro-En	$612k$	$1,999$	$1,999$
WMT15 Fi-En	$1.8M$	$1,500$	$1,370$

Table 5: Statistics of the corpora (# sentences).

Segmenter Settings For BERTSeg, we used the characterBERT model (El Boukkouri et al., 2020) trained on English Wikipedia data as encoder, and pre-processed the English data of each dataset to obtain word embeddings. Our transformer decoder was 4-layer with 1 attention head. All hidden sizes in the model were 768. The vocabulary of possible subwords used a BPE vocabulary obtained from the English part of each dataset. To prevent overfitting, we set the gradient clip to 1.0 and trained the model until the loss of $7k$ high-frequency words was stable. BERTSeg-Regularization generated 10 segmentations with the highest probability for each word and t was set to 5. We generated data of each epoch dynamically. Our method was applied to the English sentences, whereas sentences in the other languages used BPE or BPE-dropout.

Baseline methods are BPE (Sennrich et al., 2016),[1] VOLT (Xu et al., 2021),[2] DPE (He et al., 2020)[3] and BPE-dropout (Provilkov et al., 2020).[4] We used the official implementations with default settings of each method for sentences in both source and target languages.

NMT Settings We used the *transformer*$_{base}$ architecture (Vaswani et al., 2017) and the fairseq framework (Ott et al., 2019). We trained the model until no BLEU score improvement for 10 epochs on the validation set. During inference, beam size was 12 and length penalty was 1.4. We report sacreBLEU (Post, 2018) and METEOR (Banerjee and Lavie, 2005) on detokenized outputs.

4 Results and Analysis

MT Results Tables 3, 4, 6, and 7 compare the proposed methods with baseline methods. First, BERTSeg-Regularization achieves the best performance in all directions, significantly boosting BLEU scores up to 8 points and METEOR scores up to 5 points over BPE. Second, regularization is effective: methods with regularization show higher BLEU scores. Among methods w/o regularization, BERTSeg yields the highest BLEU and METEOR scores in most directions. Finally, we found the proposed method especially effective in low-resource scenarios with the help of the pre-trained model trained on large-scale data. As the train set grows, BPE and DPE gradually learn good segmentations, making the gap between BERTSeg smaller.

	IWSLT15 Vi→En	**WMT16 Ro→En**	**WMT15 Fi→En**
w/o Regularization			
BPE (Sennrich et al., 2016)	27.09	32.54	17.45
VOLT (Xu et al., 2021)	27.16	31.89	17.25
DPE (He et al., 2020)	27.40	29.95	16.14
BERTSeg	27.80$_{+0.7}$	32.33$^{\circ}_{-0.2}$	17.54$^{\circ}_{+0.1}$
With Regularization			
BPE-dropout (Provilkov et al., 2020)	28.76	33.59	**18.50**
BERTSeg-Regularization	**30.09**$^{*\circ}_{\mathbf{+3.0}}$	**33.82**$^{*\circ}_{\mathbf{+1.3}}$	**18.46**$^{*\circ}_{\mathbf{+1.0}}$

Table 6: **High-resource MT BLEU score results.** Statistical significance $p < 0.001$ is indicated by * against BPE and by $^{\circ}$ against DPE. Subscript values denote the BLEU score differences from BPE.

	IWSLT15 Vi→En	**WMT16 Ro→En**	**WMT15 Fi→En**
w/o Regularization			
BPE (Sennrich et al., 2016)	31.16	35.18	27.06
VOLT (Xu et al., 2021)	30.90	34.90	26.73
DPE (He et al., 2020)	31.07	30.15	26.00
BERTSeg	31.36$_{+0.2}$	35.16$_{-0.0}$	27.32$_{+0.3}$
With Regularization			
BPE-dropout (Provilkov et al., 2020)	32.09	35.73	28.39
BERTSeg-Regularization	**32.37**$_{\mathbf{+1.2}}$	**36.29** $_{\mathbf{+1.1}}$	**28.61**$_{\mathbf{+1.6}}$

Table 7: **High-resource MT METEOR score results.** Subscript values denote the BLEU score differences from BPE.

Training Speeds As presented in Table 8, the training speed of BERTSeg is substantially faster than the previous neural method DPE because it trains on word-level data. According to Zipf's law, the number of distinct words in a document increases much slower than the increment of the total number of words. The speed is comparable to non-neural approaches, BPE, and faster than VOLT.

	ALT	**WMT16 Ro-En**
†BPE (Sennrich et al., 2016)	4	13
†VOLT (Xu et al., 2021)	960	1,747
$^{\diamondsuit}$DPE (He et al., 2020)	3,477	68,334
$^{\spadesuit}$**BERTSeg**	58	391

Table 8: Training speeds (seconds). †: trained on CPU, $^{\diamondsuit}$: on 8 32GB GPUs, $^{\spadesuit}$ on 1 12GB GPU.

Size of Training Data With the pre-trained encoder, we can train a high-quality segmenter with a tiny train set. We train BERTSeg on words from 500k English sentences in the news commentary dataset and apply it to the ALT English words. The averaged BLEU score for MT is 24.45 whereas using only 18k ALT English data to train BERTSeg achieved 24.68 points, which are almost the same.

Subword Frequency Distribution Figure 2 shows the distribution of subword frequency in the decoded ALT train set of different methods with the same BPE vocabulary. Compared with BPE, BERTSeg generates more high-frequency (> 1000) subwords such as *ed* and *ing*. At the same time, more subwords in the vocabulary are not used during inference (with frequency 0). This phenomenon is also present in the comparison of BERTSeg-Regularization and BPE-dropout. Based on this

[1] https://github.com/google/sentencepiece
[2] https://github.com/Jingjing-NLP/VOLT
[3] https://github.com/xlhex/dpe
[4] https://github.com/google/sentencepiece

observation, it is possible to use a smaller vocabulary for BERTSeg. Additionally, we found the total subwords frequency of BERTSeg is higher because sometimes it also segments high-frequency words into subwords such as *years* into *year/s* whereas BPE keeps it as *years*.

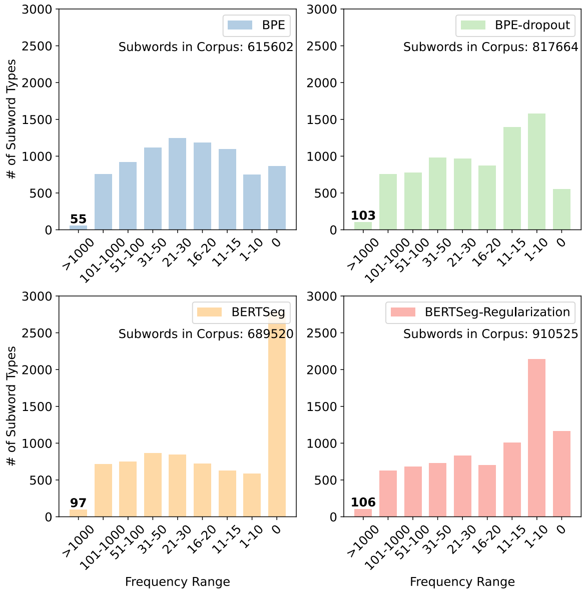


Figure 2: Subword frequency distributions of BPE, BPE-dropout, BERTSeg, and BERTSeg-Regularization.

Zero-shot Word Segmentations Table 9 demonstrates the strong generalization ability on unseen words in the test set. Different from BPE which prefers high frequency pieces such as *fell* and *hip* in the word *fellowships*, BERTSeg identifies meaningful fragments *fellow* and *ships*.

BERTSeg	**BPE** (Sennrich et al., 2016)
fellow/ships	fell/ows/hip/s
re/creation/al	rec/re/ational
dis/claim/er/s	discl/aim/ers
post/season	pos/ts/e/ason
re/fresh/ed	ref/res/hed
worse/n/s	wor/s/ens

Table 9: BERTSeg and BPE tested on unseen words.

5 Related Work

Early NMT studies apply word-level vocabulary to represent only frequent words, which causes the out-of-vocabulary (OOV) problem (Sutskever et al., 2014). To address this, character-based (Kim et al., 2016; Costa-jussà and Fonollosa, 2016; Ling et al., 2015), hybrid word-character based (Luong and Manning, 2016), or UTF-8 based (Shaham and Levy, 2021) NMT models were proposed. However, the resultant long input/output sequences increase the model and computational complexity.

Subword segmentation methods address the OOV problem by segmenting words into subwords that are in a fixed vocabulary of character n-grams. BPE (Sennrich et al., 2016; Gage, 1994) generates the subword vocabulary by first splitting all the sentences into characters, then iteratively saving the most frequent adjacent pairs into the vocabulary and merging them, until reaching the desired size. Each test sentence is segmented similarly. WordPiece (Schuster and Nakajima, 2012) and SentencePiece (Kudo and Richardson, 2018) are another two widely-used subword methods.

Among the subword methods, BPE (Sennrich et al., 2016) does not model the input sequence whereas SentencePiece (Kudo and Richardson, 2018) applies a unigram model to output probabilities of each segmentation. Based on sequence modeling via segmentations theory (Wang et al., 2017), the generation probability of a target sentence can be calculated by the sum of probabilities of all its possible segmentations. DPE (He et al., 2020) models the whole target sentence conditioned on the source sentence. However, we show that modeling words conditioned on their semantic embedding is a more efficient way.

Regularization as data augmentation can boost performance. BPE-dropout (Provilkov et al., 2020) randomly drops subword merge operation. SPM-regularization (Kudo, 2018) generates multiple segmentations with their probabilities. Leveraging the dynamic programming algorithm, we retrace the global best-*n* segmentations with the highest probabilities in polynomial time.

6 Conclusion and Future Work

We proposed BERTSeg, an unsupervised neural subword segmenter for NMT, together with a regularization algorithm. MT results showed significant improvement over frequency-based and neural network-based methods. The training is efficient even compared with non-neural methods. To address the limitations shown in Appendix A, future works include eliminating the dependency on the BPE vocabulary, extending to a multilingual segmenter with mBERT (Devlin et al., 2019) embeddings, and applying it to other generation tasks.

References

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

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

Marta R. Costa-jussà 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. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

C. M. Downey, Fei Xia, Gina-Anne Levow, and Shane Steinert-Threlkeld. 2021. A masked segmental language model for unsupervised natural language segmentation.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In Proceedings of the 28th International Conference on Computational Linguistics, pages 6903–6915, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. C Users Journal, 12(2):23–38.

Edouard Grave, Sainbayar Sukhbaatar, Piotr Bojanowski, and Armand Joulin. 2019. Training hybrid language models by marginalizing over segmentations. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1477–1482, Florence, Italy. Association for Computational Linguistics.

Xuanli He, 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. Association for Computational Linguistics.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2019. Learning to discover, ground and use words with segmental neural language models. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6429–6441, Florence, Italy. Association for Computational Linguistics.

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

Philipp Koehn, 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. Association for Computational Linguistics.

Julia Kreutzer and Artem Sokolov. 2018. Learning to segment inputs for nmt favors character-level processing.

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

Taku Kudo 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. Association for Computational Linguistics.

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

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

Christopher Manning, 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. Association for Computational Linguistics.

Myle Ott, 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. Association for Computational Linguistics.

Matt Post. 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. Association for Computational Linguistics.

Ivan Provilkov, 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. Association for Computational Linguistics.

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

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

Uri Shaham 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. Association for Computational Linguistics.

Zhiqing Sun and Zhi-Hong Deng. 2018. Unsupervised neural word segmentation for Chinese via segmental language modeling. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4915–4920, Brussels, Belgium. Association for Computational Linguistics.

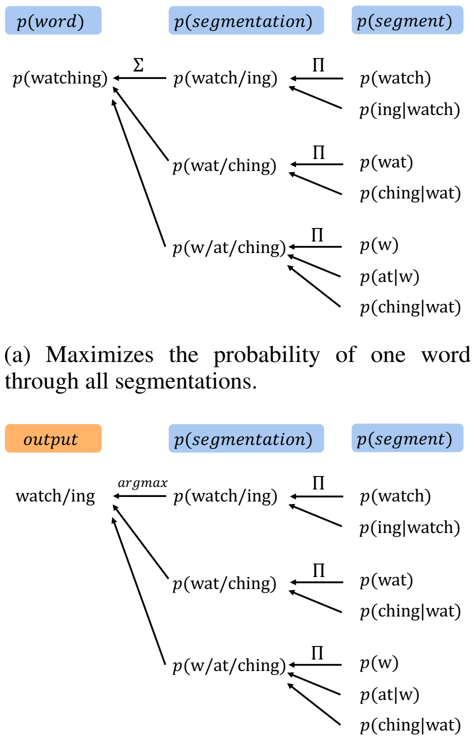
Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 3104–3112. Curran Associates, Inc.

Arseny Tolmachev, 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. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc.

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

Jingjing Xu, 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. Association for Computational Linguistics.



(a) Maximizes the probability of one word through all segmentations.

(b) Retrace the optimal segmentation with the highest probability.

Figure 3: An example of the training and inference phases.

A Limitations

Despite the effectiveness and efficiency, the proposed method has the following methodological and experimental limitations ranked in order of importance. We also provide directions to solve them as future works.

Dependency on BPE Vocabulary BERTSeg is a model to learn optimal segmentations for words but not paired with a vocabulary generation algorithm. Currently, the vocabulary is generated by BPE, therefore, many subwords in the vocabulary are not used, as shown in Figure 2. It is possible to address this by first generating a large vocabulary and then shrinking it iteratively, saving the commonly used subwords only, motivated by the SentencePiece work (Kudo and Richardson, 2018).

Target Side Only The goal of BERTSeg is to maximize the generation probability as shown in Eq. (2), therefore, can only apply to the target side data in generation tasks. Applying BERTSeg to the source side data will not improve the MT performance in our preliminary experiments, which is also reported in the DPE work (He et al., 2020). To address this, a dual segmenter model is needed to optimize both the target segmentations and source segmentations.

English Subword Segmenter Only Currently we only train the subword segmenter for English due to there is only an English characterBERT model. However, we believe using embeddings from BERT or mBERT will not affect the performance, although it adds a dependency on the BERT tokenizer. To extend BERTSeg to mBERTSeg, a multilingual characterBERT is needed.

Definition of Good Segmentation The definition of good subword segmentation is beyond the scope of this paper, and we use the BLEU score as the metric to measure downstream tasks performance. However, measuring the segmentation quality is a more direct way. To achieve this, crowd-sourcing is a promising way to obtain a supervised subword segmentation dataset, at least for frequent words.

B Example: Training and Inference

The training and inference are given by Equations 2 and 3, respectively. They are based on the sequence modeling theory that is first introduced in Wang et al. (2017) and there are multiple applications (Kawakami et al., 2019; Sun and Deng, 2018; Downey et al., 2021; Grave et al., 2019; Kreutzer and Sokolov, 2018; Wang et al., 2017). To understand the unsupervised training and inference processes more intuitively, we provide an example as illustrated in Figure 3.

In the training phase, the probability of the word "watching" is calculated by summing all possible segmentations. In the inference phase, we retrace the segmentation with the maximum probability for BERTSeg and retrace the best N segmentations for BERTSeg-Regularization.

We also attached the code and will make the code public for better understanding and reproduction.

C Example: Segmentations

We provide examples comparing the proposed method with BPE including high-frequency words, rare words and unseen words as shown in Table 10. We have the following observations:

- **For frequent words**, BERTSeg sometimes segment them into subwords even the word is in the vocabulary such as *official/s* and *use/d*. Additionally, the model can discriminate the

BERTSeg	BPE
Frequent words	
official/s	officials
edit/ion	edition
use/d	used
farm/er/s	far/mers
contribute/d	contrib/uted
normal/ly	norm/ally
seven/th	sevent/h
challenge/d	challeng/ed
over/night	o/vern/ight
language/s	langu/ages
Rare words	
inter/face/s	inter/f/aces
sea/side	se/as/ide
ab/normal/ly	ab/n/orm/ally
b/y/stand/er	by/st/ander
dis/comfort	disc/om/fort
un/warrant/ed	un/w/arr/anted
in/definitely	ind/ef/in/itely
Unseen words	
stable/d	st/ab/led
save/r/s	sa/vers
M/illion/s	Mill/ions
Free/way	Fre/ew/ay
M/i/s/behavior	M/is/be/hav/ior
m/o/u/r/n/ed	m/our/ned
M/a/d/a/m/e	Mad/ame

Table 10: BERTSeg and BPE segmentations on frequent words, rare words and unseen words.

ambiguous situations very well. For example, the model can extract the prototype *challenge* from the word *challenged*.

- **For rare words** with frequency < 5 in the training set, BERTSeg gives much better segmentations than BPE, because BPE is a frequency-based method and thus handles rare words poorly.
- **For unseen words**, although the BERTSeg model gives better segmentations than BPE, we found that sometimes it oversegments words such as *M/a/d/a/m/e*. We guess it's due to the low-quality word embedding from characterBERT, and we do not know the impact of this on the MT results.

D Implementation Details of Baselines

This section aims to help to reproduce the results in the paper more easily. In the meantime, we provide some observations from the experiments.

D.1 BPE

Vocabulary Size Vocabulary size is a very important hyperparameter for the NMT experiments. For the ALT dataset, we did hyperparameter searching and $8,000$ gave the highest BLEU scores averaged in all directions. For the IWSLT15 Vi-En, WMT16 Ro-En and WMT15 Fi-En datasets, we have tried two settings: $8,000$ and $32,000$, where using $8,000$ gave a higher performance.

The Size of Monolingual Data In low-resource scenarios, using a larger monolingual dataset in the same domain to generate the BPE vocabulary gives better performance. We have used 500k English monolingual data from the news commentary dataset, and it gives 0.4 BLUE score improvements over using 18k ALT data to generate the BPE vocabulary.

Comparison with SentencePiece We used BPE as the baseline method because it gave higher performance (about 0.2 BLEU scores) than SentencePiece in low-resource scenarios. We assume that in the situation where the sentence is tokenized into words, the performance of BPE will be higher because the subwords in the BPE vocabulary do not contain spaces.

D.2 VOLT

Dataset	Language	Size
ALT	En/Id/Ja	8k
ALT	Ms	6k
ALT	Vi	7k
ALT	Fil/Zh	9k
IWSLT15 Vi-En	En/Vi	7k
WMT16 Ro-En	En	10k
WMT16 Ro-En	Ro	11k
WMT15 Fi-En	En	10k
WMT15 Fi-En	Fi	8k

Table 11: Optimal BPE vocabulary sizes of languages in each dataset.

Table 11 illustrates the optimal sizes of BPE vocabularies of each dataset calculated by the VOLT algorithm. The optimal numbers are very similar to the results we got from hyperparameter searching, showing the effectiveness of the VOLT algorithm.

D.3 BPE-dropout

We have tried BPE-dropout rates of 0.05 and 0.1, where 0.1 gave higher BLEU scores. Note that statical BPE-dropout is not helpful, it is necessary to segment the train set for each epoch.

D.4 DPE

We basically followed the official implementations. The training requires 8 32GB GPUs to train for about one week for large datasets.