# MaxMatch-Dropout: Subword Regularization for WordPiece

**Tatsuya Hiraoka**
Tokyo Institute of Technology*
hiraoka.tatsuya@fujitsu.com

## Abstract

We present a subword regularization method for WordPiece, which uses a maximum matching algorithm for tokenization. The proposed method, MaxMatch-Dropout, randomly drops words in a search using the maximum matching algorithm. It realizes finetuning with subword regularization for popular pretrained language models such as BERT-base. The experimental results demonstrate that MaxMatch-Dropout improves the performance of text classification and machine translation tasks as well as other subword regularization methods. Moreover, we provide a comparative analysis of subword regularization methods: subword regularization with SentencePiece (Unigram), BPE-Dropout, and MaxMatch-Dropout.

## 1 Introduction

Subword regularization (Kudo, 2018) is a well-known technique for improving the performance of NLP systems, whereby a model is trained with various tokenizations that are sampled for each training epoch. This approach provides data augmentation and model robustness against tokenization differences.

Kudo (2018) first introduced subword regularization using a unigram language model that was included in their tokenization tool, namely SentencePiece (Kudo and Richardson, 2018), and reported its effectiveness on machine translation tasks. Provilkov et al. (2020) proposed a subword regularization method for byte pair encoding (BPE) known as BPE-Dropout and demonstrated the superiority of their method over that using the unigram language model in machine translation tasks. Moreover, subword regularization contributes to the performance improvement of text classification tasks (Hiraoka et al., 2019).

*The author is currently affiliated with Fujitsu Limited. This work was carried out at the Tokyo Institute of Technology.

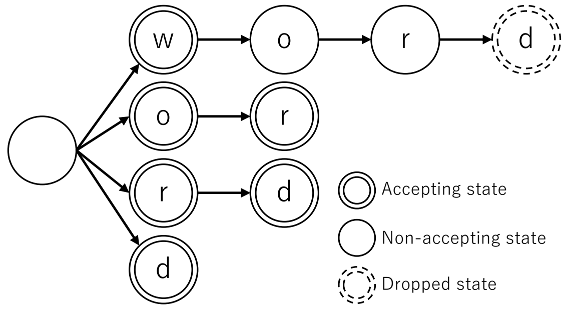


Figure 1: MaxMatch-Dropout randomly removes accepting states in the trie. In this figure, a state corresponding to “word” is dropped and a single input “word” is tokenized as “w, or, d.”

As subword regularization is implemented as a modification of a tokenizer, each method is specialized to a particular tokenizer type. For example, the original subword regularization (Kudo, 2018) is specialized to a tokenizer that uses the unigram language model and BPE-Dropout is specialized to the BPE-based tokenizer. However, these existing subword regularization tools cannot be directly applied to the other common tokenizers such as WordPiece (Song et al., 2021).

WordPiece is a tokenizer that is based on the maximum matching algorithm. It is used as the default tokenizer for the popular pretrained language model BERT (Devlin et al., 2018). Although the widely used BERT models (e.g., BERT-base) can improve the performance of various NLP tasks, subword regularization cannot be used for the finetuning of the model because no subword regularization method exists for WordPiece. The use of subword regularization for the finetuning of pretrained models with WordPiece may result in a further performance improvement.

In this paper, we present a simple modification of WordPiece for the use of subword regularization. The proposed method, which is known as MaxMatch-Dropout, randomly drops words in a vocabulary during the tokenization process. That



*Proceedings of the 29th International Conference on Computational Linguistics*, pages 4864–4872
October 12–17, 2022.

**Algorithm 1** Algorithm for Word Tokenization

**Require:** Single Word $w$, Vocabulary $V$, Dropout Rate $q$.
$S \leftarrow$ Empty List
Index of Characters $i \leftarrow 1$
**while** $i < |w|$ **do**
  Subword $s \leftarrow \varnothing$
  **for** $j = 1$ to $|w| - i$ **do**
    **if** $w_{i:i+j} \in V$ and $\mathrm{Ber}(1-q)$ **then**
      $s \leftarrow w_{i:i+j}$
  **if** $s = \varnothing$ **then return** [UNK]
  **else**
    Add $s$ to $S$
    $i \leftarrow i + |s|$
**return** $S$

is, MaxMatch-Dropout randomly removes accepting states from a trie for tokenization. The experimental results demonstrate that MaxMatch-Dropout improves the performance of text classification and machine translation in several languages, as well as other subword regularization methods. Furthermore, MaxMatch-Dropout contributes to a further performance improvement with pretrained BERT on text classification in English, Korean, and Japanese.

## 2 Maximum Matching

A simple modification to the maximum matching algorithm is implemented so that MaxMatch-Dropout can realize subword regularization. Prior to explaining the modification, we briefly review the maximum matching on which the proposed method is based[1].

Given a vocabulary and a single word, the maximum matching searches the longest subword in the vocabulary and greedily tokenizes the word into a sequence of subwords from beginning to end. For example, let the vocabulary be composed of {a, b, c, d, abc, bcd}. The tokenizer with the maximum matching divides a word “abcd” into “abc, d”[2]. As the maximum matching searches subwords from the beginning of the word, this word is not tokenized as “a, bcd.” When an input word includes an unknown character, such as “abce,” the tokenizer replaces this word with a special token, “[UNK].” This tokenization process is usually implemented using a trie. The detailed tokenization process using the maximum matching for this example with the trie (Figure 4) is described in Appendix A.

[1] Song et al. (2021) explains the efficient implementation of the maximum matching in detail.

[2] We do not use special tokens for a subword that begins in the middle of a word (e.g., “##”) for simple explanation.

## 3 Proposed Method: MaxMatch-Dropout

The proposed method extends the maximum matching with an additional dropout process. This method randomly replaces accepting states into non-accepting states with **dropped states**. That is, accepting tokens are randomly skipped with a specified probability $q$, where $q$ is a hyperparameter.

Figure 1 depicts the tokenization process of a word “word” with a vocabulary that includes {w, o, r, d, or, rd, word}. Although the maximum matched subword beginning with the first character is “word” in the vocabulary, in this case, the state corresponding to “word” is dropped. Thus, the latest accepted subword “w” is yielded and the next matching begins from the second character. Finally, the tokenization process results in “w, or, d.”

This process is also outlined in Algorithm 1 [3]. In the algorithm, $w_{i,i+j}$ denotes a subword beginning from the $i$-th character and ending with the $(i+j-1)$-th character in the word $w$, where $|w|$ and $|s|$ are the lengths of the input word and subword, respectively. Moreover, $Ber(1-q)$ denotes a Bernoulli distribution that returns 1 with a probability of $1-q$.

The tokenization process of MaxMatch-Dropout is detailed in Table 6 of Appendix A. The difference between MaxMatch-Dropout and the original maximum matching can be observed by comparing Tables 5 and 6.

The regularization strength can be tuned using the hyperparameter $q$. The proposed method is equivalent to the original maximum matching with $q = 0.0$, and it tokenizes a word into characters with $q = 1.0$ if all characters are included in the vocabulary.

The official code is available at https://github.com/tatHi/maxmatch_dropout.

## 4 Experiments

We conducted experiments on text classification and machine translation tasks to validate the performance improvement provided by MaxMatch-Dropout.

We used two tokenizers and subword regularization methods as a reference for both tasks: SentencePiece (Unigram) (Kudo and Richardson, 2018) with subword regularization (Sub. Reg.) (Kudo,

[3] Algorithm 1 does not use a trie for simple explanation.

| | English | | | | | | | Korean | | | Japanese | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | APG | APR | TS | QNLI | QQP | RTE | SST-2 | NLI | STS | YNAT | TR | WRIME |
| $\lvert V \rvert$ | 32K | 32K | 32K | 32K | 32K | 12K | 8K | 24K | 16K | 32K | 16K | 12K |
| Metric | F1 | F1 | F1 | Acc. | F1 | Acc. | Acc. | Acc. | F1 | F1 | F1 | F1 |
| *BiLSTM* | | | | | | | | | | | | |
| Unigram | 69.05 | 65.85 | 76.21 | 66.48 | 83.61 | 49.10 | 80.05 | 41.93 | 67.02 | 68.57 | 86.6 | 46.36 |
| + Sub. Reg. | **70.65** | **66.80** | **77.49** | **66.56** | **83.91** | **53.31** | **83.30** | **42.84** | **68.08** | **73.67** | **87.11** | **49.47** |
| BPE | 67.10 | 64.67 | 75.24 | **67.11** | **82.82** | 53.07 | 78.10 | 41.22 | **67.42** | 64.27 | 84.95 | 44.34 |
| + BPE-Dropout | **68.45** | **65.38** | **76.04** | 66.69 | 82.69 | **53.97** | **82.00** | **41.52** | 66.26 | **69.12** | **85.68** | **46.01** |
| WordPiece | 63.17 | 62.97 | 73.14 | 64.04 | 82.11 | 53.55 | 81.04 | 39.96 | 61.75 | 62.44 | 84.95 | 46.36 |
| + MM-Dropout | **64.90** | **64.36** | **75.22** | **64.28** | **82.14** | **53.91** | **83.75** | **40.61** | **62.88** | **70.08** | **86.98** | **47.28** |
| *BERT* | | | | | | | | | | | | |
| WordPiece | 77.28 | 70.99 | 81.93 | 89.45 | 89.83 | 62.00 | 90.97 | 82.18 | 83.22 | 83.96 | 89.08 | 89.08 |
| + MM-Dropout | **78.55** | **71.68** | **82.08** | **89.74** | **89.86** | **62.27** | **91.07** | **82.19** | **85.43** | **84.31** | **89.14** | **89.14** |

Table 1: Experimental results of text classification (averaged scores of five runs). The higher scores for the tokenizations with/without subword regularization are indicated in bold. The scores that significantly surpassed the results without subword regularization ($p < 0.05$, McNemar's test) are underlined.

2018) and BPE (Sennrich et al., 2016) with BPE-Dropout (Provilkov et al., 2020). We employed WordPiece (Song et al., 2021), which was implemented by HuggingFace (Wolf et al., 2020), as a basic tokenizer for the proposed MaxMatch-Dropout [4].

We set the vocabulary size of each tokenizer to be equal to compare the three methods as fairly as possible. The vocabulary of each tokenizer included all characters that appeared in the training splits. We selected the hyperparameters for the subword regularization (e.g., $q$ of MaxMatch-Dropout) according to the performance on the development splits. Note that we could not fairly compare the performance of MaxMatch-Dropout to that of other subword regularization methods because they are based on different tokenizers and vocabularies. WordPiece was used as the baseline for MaxMatch-Dropout to investigate whether the method could successfully perform subword regularization and improve the performance similarly to other methods.

### 4.1 Text Classification

**Datasets** We exploited text classification datasets in three languages: English, Korean, and Japanese. **APG** and **APR** are genre prediction and rating prediction, respectively, on review texts that were created from the Amazon Product Dataset (He and McAuley, 2016). **TS** is a sentiment classification for tweets [5]. We also employed **QNLI** (Rajpurkar et al., 2016), **QQP** (Chen et al., 2018), **RTE** (Bentivogli et al.), and **SST-2** (Socher et al., 2013) from the GLUE benchmark (Wang et al., 2018). **NLI**, **STS**, and **YNAT** are text classification datasets that are included in Korean GLUE (KLUE) (Park et al., 2021). **TR** (Suzuki, 2019) and **WRIME** (Kajiwara et al., 2021) are sentiment classification datasets for tweets in Japanese. We used the original development sets as test sets and exploited a randomly selected 10% of the original training sets as development sets for the datasets in GLUE and KLUE owing to the numerous experimental trials.

**Setup** We used two backbones for the text classification: BiLSTM (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005) and BERT (Devlin et al., 2018). We employed BERT-base-cased[6], BERT-kor-base[7](Kim, 2020), and BERT-base-Japanese-v2[8] for the English, Korean, and Japanese datasets, respectively. All of these BERT models employ WordPiece as their tokenizers, and we finetuned them using MaxMatch-Dropout. We set the maximum number of training epochs to 20 for BiLSTM and the finetuning epochs to 5 for BERT. The trained model with the highest score in the development split was selected and evaluated on the test split. We selected the vocabulary sizes according to the performance on the development splits when using WordPiece without MaxMatch-Dropout. The selected vocabulary sizes were applied to all tokenizers.

**Results** Table 1 presents the experimental results for the text classification. The table demon-

[4] Table 12 in the Appendix presents tokenization examples for each tokenizer.

[5] `https://www.kaggle.com/c/twitter-sentiment-analysis2`

[6] `https://huggingface.co/bert-base-cased`

[7] `https://huggingface.co/kykim/bert-kor-base`

[8] `https://huggingface.co/cl-tohoku/bert-base-japanese-v2`

| | IWSLT14 | | IWSLT15 | | | |
|---|---|---|---|---|---|---|
| | DeEn | EnDe | ViEn | EnVi | ZhEn | EnZh |
| Unigram | 36.55 | 27.89 | 30.28 | 29.39 | 22.64 | 20.55 |
| + Sub. Reg. | **38.50** | **29.45** | **31.58** | **30.96** | **23.81** | **21.79** |
| BPE | 35.77 | 27.87 | 30.05 | 29.25 | 18.80 | 20.61 |
| + BPE-Dropout | **37.81** | **29.15** | **31.39** | **31.23** | **20.67** | **22.02** |
| WordPiece | 36.22 | 27.58 | 30.13 | 29.40 | 17.24 | 20.45 |
| + MM-Dropout | **38.30** | **29.54** | **31.71** | **31.14** | **18.21** | **21.55** |

Table 2: Experimental results of machine translation (averaged scores of three runs). ScareBLEU (Post, 2018) was used as the metric. Scores that significantly surpassed the results without subword regularization ($p < 0.05$, bootstrap resampling (Koehn et al., 2007)) are underlined.

strates that MaxMatch-Dropout (MM-Dropout) improved the performance as well as the other subword regularization methods. In addition to the improvement in the BiLSTM-based classifiers, MaxMatch-Dropout enhanced the performance of the BERT-based classifiers. These results indicate that MaxMatch-Dropout is a useful subword regularization method for WordPiece as well as effective for BERT.

### 4.2 Machine Translation

**Datasets** We employed three language pairs for the machine translation tasks: the De-En, Vi-En, and Zh-En pairs from the IWSLT corpora. We selected these datasets because subword regularization is particularly efficient in low-resource environments (Kudo, 2018; Hiraoka et al., 2021; Takase et al., 2022).

**Setup** We applied the Transformer (Vaswani et al., 2017), which was implemented by Fairseq (Ott et al., 2019), for the IWSLT settings. We trained the model with 100 epochs and averaged the parameters of the final 10 epochs. We evaluated the performance on the Chinese dataset using character-level BLEU. Following Provilkov et al. (2020), we set the vocabulary size to 4K for English, German, and Vietnamese, and 16K for Chinese.

**Results** Table 2 displays the experimental results for the machine translation. The table demonstrates that MaxMatch-Dropout improved the performance in all language pairs. The results indicate that the proposed method is effective for machine translation as well as existing subword regularization methods.

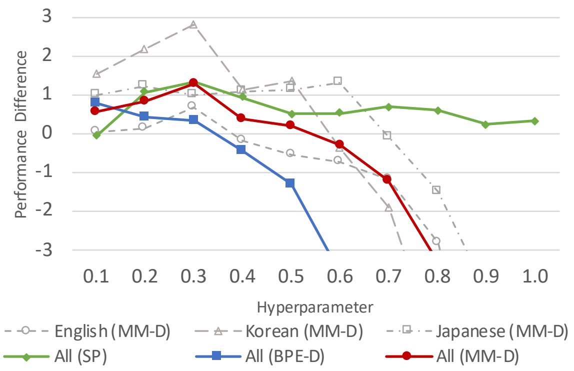


Figure 2: Performance differences with and without subword regularization against hyperparameters and for different languages on text classification datasets. MM-D, SP, and BPE-D denote MaxMatch-Dropout, SentencePiece (Unigram), and BPE-Dropout, respectively.

## 5 Discussion

### 5.1 Effect of Hyperparameters

Figure 2 depicts the averaged performance improvement over several text classification datasets against different hyperparameters. The figure indicates that the subword regularization of SentencePiece (Unigram) was the most robust against the hyperparameters among the three methods. Although both BPE-Dropout and MaxMatch-Dropout could realize subword regularization using the dropout technique for the tokenization strategy, MaxMatch-Dropout was more robust against the hyperparameters than BPE-Dropout. This result demonstrates that a performance improvement can be achieved in WordPiece-based systems using MaxMatch-Dropout with approximately selected hyperparameters (e.g., $q < 0.5$).

Figure 2 also shows the averaged performance on the datasets in each language against the hyperparameters of MaxMatch-Dropout (dashed lines). It can be observed that MaxMatch-Dropout was more effective for Asian languages than English. It is considered that this is because Korean and Japanese contain various types of n-grams and many tokenization candidates exist for a single sentence compared to English.

### 5.2 Token Length

In this subsection, we analyze the token length in the sampled tokenizations. We sampled the tokenization of the training dataset (APG) with three subword regularization methods and counted the token lengths for 10 trials.

Figure 3 presents the frequency of token lengths

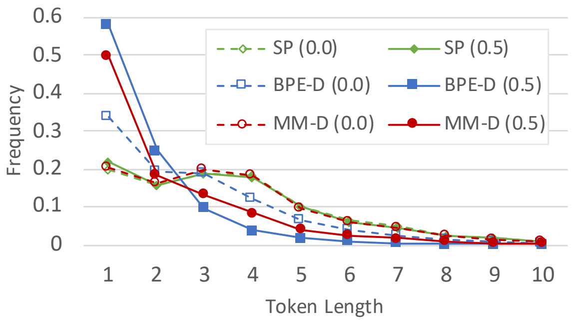


Figure 3: Frequency of token lengths with each subword regularization method on APG dataset (English). 0.0 denotes the vanilla settings without subword regularization. 0.5 indicates subword regularization when the hyperparameter was 0.5 (e.g., $q = 0.5$). MM-D, SP, and BPE-D denote MaxMatch-Dropout, SentencePiece (Unigram), and BPE-Dropout, respectively.

in the tokenized training datasets with/without subword regularization. The figure indicates that the length frequency did not change, regardless of the use of subword regularization, when SentencePiece (Unigram) was applied. In contrast, both MaxMatch-Dropout (MM-D) and BPE-Dropout (BPE-D) yielded many characters when the hyperparameter was 0.5, because they are based on the token-level dropout and yield characters when the hyperparameter is 1.0. However, the frequency curve of MaxMatch-Dropout was gentler than that of BPE-Dropout. We believe that this tendency aided in the robustness of the MaxMatch-Dropout performance, as reported in Section 5.1.

## 6 Conclusion

We have introduced a subword regularization method for WordPiece, which is a common tokenizer for BERT. The proposed method, MaxMatch-Dropout, modifies the tokenization process using the maximum matching to drop words in the vocabulary randomly. This simple modification can realize subword regularization for WordPiece. Furthermore, the experimental results demonstrated that MaxMatch-Dropout can improve the performance of BERT. MaxMatch-Dropout is also effective in the training of text classification tasks without BERT and machine translation tasks, as well as existing subword regularization methods.

## Acknowledgement

This work was supported by JST, ACT-X Grant Number JPMJAX21AM, Japan.

## References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2018. Quora question pairs. *University of Waterloo*, pages 1–7.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052. IEEE.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.

Tatsuya Hiraoka, Hiroyuki Shindo, and Yuji Matsumoto. 2019. Stochastic tokenization with a language model for neural text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1620–1629.

Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Atsushi Keyaki, and Naoaki Okazaki. 2021. Joint optimization of tokenization and downstream model. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 244–255.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Tomoyuki Kajiwara, Chenhui Chu, Noriko Takemura, Yuta Nakashima, and Hajime Nagahara. 2021. WRIME: A new dataset for emotional intensity estimation with subjective and objective annotations. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2095–2104, Online. Association for Computational Linguistics.

Kiyoung Kim. 2020. Pretrained language models for korean. https://github.com/kiyoungkim1/LMkor.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

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.

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.

Xing Niu, Prashant Mathur, Georgiana Dinu, and Yaser Al-Onaizan. 2020. Evaluating robustness to input perturbations for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8538–8544.

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 NAACL-HLT 2019: Demonstrations*.

Sungjoon Park, Jihyung Moon, Sungdong Kim, Won Ik Cho, Ji Yoon Han, Jangwon Park, Chisung Song, Junseong Kim, Youngsook Song, Taehwan Oh, et al. 2021. Klue: Korean language understanding evaluation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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, Brussels, Belgium. 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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE.

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)*, volume 1, pages P1715–1725.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. 2021. Fast wordpiece tokenization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2089–2103.

Yu Suzuki. 2019. Filtering method for twitter streaming data using human-in-the-loop machine learning. *Journal of Information Processing*, 27:404–410.

Sho Takase, Tatsuya Hiraoka, and Naoaki Okazaki. 2022. Single model ensemble for subword regularized models in low-resource machine translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2536–2541.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

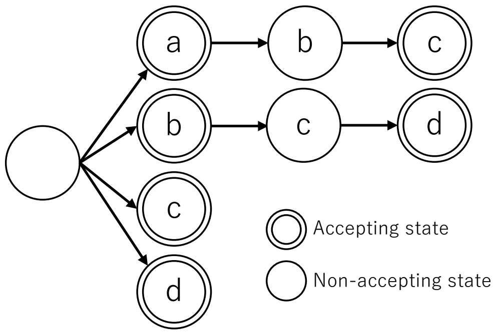


Figure 4: Trie for vocabulary including tokens {a, b, c, d, abc, bcd}.

| Read | Action | Output |
|---|---|---|
| a | Accept "a" | |
| b | Non-accept "ab" | |
| c | Accept "abc" | |
| d | Reject the transition to "abcd"<br>& Yield the latest subword | abc |
| d | Accept "d" | |
| $ | Reject the transition to "d$"<br>& Yield the latest subword | abc, d |

Table 3: Operation for tokenizing input word "abcd" into "abc, d" using trie shown in Figure 4. "$" denotes a special symbol indicating the end of the word.

| Read | Action | Output |
|---|---|---|
| a | Accept "a" | |
| b | Non-accept "ab" | |
| c | Accept "abc" | |
| e | Reject the transition to "abcd"<br>& Yield the latest subword | abc |
| e | Detect an OOV character<br>& Output [UNK] | [UNK] |

Table 4: Operation for tokenizing input word "abce" including out-of-vocabulary (OOV) character into "[UNK]" using trie shown in Figure 4.

| Read | Action | Output |
|---|---|---|
| w | Accept "w" | |
| o | Non-accept "wo" | |
| r | Non-accept "wor" | |
| d | Accept "word" | |
| $ | Reject the transition to "word$"<br>& Yield the latest subword | word |

Table 5: Operation for tokenizing input word "word" by applying original maximum matching (i.e., the operation without any dropout process) for trie shown in Figure 1. "$" denotes a special symbol indicating the end of the word.

## A Maximum Matching in Detail

As described in Section 2, a trie is generally used to tokenize an input word with the maximum matching algorithm. Figure 4 depicts the trie corresponding to the vocabulary that includes six tokens: {a, b, c, d, abc, bcd}. The tokenization process using this trie for the input words "abcd" and "abce" is presented in Tables 3 and 4, respectively.

Table 6 details the operation for tokenizing an input word "word" into "w, or, d" using the proposed MaxMatch-Dropout, as outlined in Section 3. Table 5 describes the tokenization process using the original maximum matching for Figure 1 without the dropout process. Therefore, the difference in the tokenization process between the original maximum matching and MaxMatch-Dropout can be observed by comparing Tables 5 and 6.

## B Related Work

This work is related to tokenization methods, which split raw texts into a sequence of tokens. Three well-known tokenization methods have been employed in recent NLP systems: SentencePiece (Unigram) (Kudo and Richardson, 2018), BPE (Sennrich et al., 2016), and WordPiece (Song et al., 2021). SentencePiece (Unigram) is a unigram language model-based tokenizer, whereas BPE employs a frequency-based tokenization technique. Although both methods are used extensively in many NLP systems, Bostrom and Durrett (2020) reported that the unigram language model-based tokenizer (i.e., SentencePiece (Unigram)) is superior to BPE in several downstream tasks. Our experimental results in Tables 1 and 2 also support this finding.

WordPiece[9] is another famous tokenizer that is mainly employed by large pretrained models such as BERT (Devlin et al., 2018). As WordPiece is based on the maximum matching algorithm, it is superior to other tokenization methods in terms of the tokenization speed. In fact, WordPiece is employed in real NLP systems such as Google searching (Song et al., 2021). However, the experimental results in this study (Table 1 and 2) demonstrated that WordPiece is inferior to SentencePiece (Unigram) and BPE in terms of performance. The proposed method can compensate for this shortcoming without decreasing the inference speed.

Kudo (2018) introduced a subword regularization technique for SentencePiece (Unigram) using dynamic programming. Provilkov et al. (2020) proposed a subword regularization method for BPE using the dropout technique. Niu et al. (2020) inves-

[9] Although the original term "wordpiece" indicates BPE-based tokenization (Schuster and Nakajima, 2012), in this paper, "WordPiece" indicates a tokenizer with the maximum matching for BERT following Song et al. (2021).

| Read | Action | Output |
|---|---|---|
| w | Accept "w" | |
| o | Non-accept "wo" | |
| r | Non-accept "wor" | |
| d | **(Randomly) Non-accept "word"** | |
| $ | Reject the transition to "word$" | |
| | & Yield the latest subword | w |
| o | Accept "o" | |
| r | Accept "or" | |
| d | Reject the transition to "ord" | |
| | & Yield the latest subword | w, or |
| d | Accept "d" | |
| $ | Reject the transition to "d$" | |
| | & Yield the latest subword | w, or, d |

Table 6: Operation for tokenizing input "word" using trie for MaxMatch-Dropout shown in Figure 1. "$" denotes a special symbol indicating the end of the word.

tigates these two methods in machine translation. This study has introduced a subword regularization method for WordPiece, and presented an in-depth investigation of the three methods in text classification and machine translation.

## C Contributions

This study contributes to the NLP community in terms of the following two main points:

- A subword regularization method for WordPiece is proposed, which improves the text classification and machine translation performance.
- An intensive performance investigation of the three famous tokenization and subword regularization methods used in NLP (i.e., SentencePiece (Unigram), BPE, and WordPiece with subword regularization) is presented.

## D Dataset Statistics

Table 7 displays the detailed information of the datasets. We report the numbers of samples in the training, development, and test splits. Furthermore, we present the number of label types for text classification datasets.

## E Detailed Experimental Settings

Tables 8 and 9 present the detailed settings of the backbone models that were used in text classification and machine translation tasks, respectively. We used the default values of PyTorch for the hyperparameters that are not described in these tables. We set the number of tokenization candidates to $\infty$ for the subword regularization of SentencePiece (Unigram).

| Dataset | Train | Dev. | Test | Labels |
|---|---|---|---|---|
| *English Text Classification* | | | | |
| APG | 96,000 | 12,000 | 12,000 | 24 |
| APR | 96,000 | 12,000 | 12,000 | 5 |
| TS | 80,000 | 10,000 | 10,000 | 2 |
| QNLI | 188,536 | 10,475 | 5,463 | 2 |
| QQP | 327,461 | 36,385 | 40,430 | 2 |
| RTE | 2,241 | 249 | 277 | 2 |
| SST-2 | 60,614 | 6,735 | 872 | 2 |
| *Korean Text Classification* | | | | |
| NLI | 22,498 | 2,500 | 3,000 | 3 |
| STS | 10,501 | 1,167 | 519 | 2 |
| YNAT | 41,110 | 4,568 | 9,107 | 7 |
| *Japanese Text Classification* | | | | |
| TR | 129,747 | 16,218 | 16,219 | 3 |
| WRIME | 30,000 | 2,500 | 2,500 | 5 |
| *Machine Translation* | | | | |
| DeEn | 160,239 | 7,283 | 6,750 | - |
| ViEn | 130,933 | 768. | 1,268 | - |
| ZhEn | 209,941 | 887. | 1,261 | - |

Table 7: Statistics of datasets.

| Parameter | BiLSTM | BERT |
|---|---|---|
| Embedding Size | 64 | 768 |
| BiLSTM/BERT Hiden Size | 256 | 768 |
| # of BiLSTM/BERT Layers | 1 | 12 |
| Dropout Rate | 0.5 | 0.1 |
| Optimizer | Adam | AdamW |
| Learning Rate | 0.001 | 0.00002 |

Table 8: Overview of hyperparameters for backbone models of text classification tasks.

We selected the hyperparameters for the subword regularization methods (the smoothing parameter for SentencePiece (Unigram) and the dropout probabilities for BPE-Dropout and MaxMatch-Dropout) according to the performance on the development splits in the experiments. Tables 10 and 11 summarize the selected values of the hyperparameters for the text classification and machine translation, respectively. Note that the other methods without subword regularization (Unigram, BPE, and WordPiece) do not require these hyperparameters.

| Parameter | Transformer |
|---|---|
| Enc/Dec Embedding Size | 512 |
| Enc/Dec FFN Embedding Size | 1,024 |
| # of Enc/Dec Attention Heads | 4 |
| # of Enc/Dec Layers | 6 |
| Clipping Norm | 0.0 |
| Dropout Rate | 0.3 |
| Weight Decay | 0.0001 |
| Max Tokens for Mini-Batch | 1,000 |
| Optimizer | Adam |
| $\beta_1$ and $\beta_2$ for Adam | 0.9, 0.98 |
| Learning Rate | 0.0005 |
| Learning Rate Scheduler | Inverse Square Root |
| Warming-Up Updates | 4,000 |

Table 9: Overview of hyperparameters for backbone model of machine translation tasks.

| | English | | | | | | | Korean | | | Japanese | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | APG | APR | TS | QNLI | QQP | RTE | SST-2 | NLI | STS | YNAT | TR | WRIME |
| *BiLSTM* | | | | | | | | | | | | |
| Unigram+Sub. Reg. | 0.2 | 0.2 | 0.2 | 0.6 | 0.9 | 0.3 | 0.2 | 0.9 | 0.3 | 0.3 | 0.4 | 1.0 |
| BPE-dropout | 0.2 | 0.2 | 0.4 | 0.1 | 0.1 | 0.1 | 0.3 | 0.3 | 0.2 | 0.3 | 0.5 | 0.2 |
| MaxMatch-dropout | 0.2 | 0.3 | 0.6 | 0.1 | 0.1 | 0.3 | 0.4 | 0.4 | 0.2 | 0.3 | 0.4 | 0.6 |
| *BERT* | | | | | | | | | | | | |
| MaxMatch-Dropout | 0.6 | 0.4 | 0.2 | 0.1 | 0.1 | 0.1 | 0.3 | 0.5 | 0.4 | 0.5 | 0.4 | 0.5 |

Table 10: Selected hyperparameters for subword regularization methods in text classification tasks.

| | IWSLT14 | | IWSLT15 | | | |
|---|---|---|---|---|---|---|
| | DeEn | EnDe | ViEn | EnVi | ZhEn | EnZh |
| Unigram + Sub. Reg. | 0.3 | 0.3 | 0.4 | 0.3 | 0.2 | 0.2 |
| BPE-Dropout | 0.1 | 0.2 | 0.2 | 0.2 | 0.3 | 0.2 |
| MaxMatch-Dropout | 0.3 | 0.3 | 0.4 | 0.1 | 0.1 | 0.2 |

Table 11: Selected hyperparameters for subword regularization methods in machine translation tasks. The selected hyperparameters were used for the subword regularization of both the source and target languages.

| Hyperparameter | Trial | Unigram+Sub. Reg. | BPE-Dropout | MaxMatch-Dropout |
|---|---|---|---|---|
| No regularization | - | characteristics | characteristics | characteristics |
| 0.1 | 1 | **character_i_s_t_ic_s** | characteristics | **characteristic_s** |
| | 2 | **character_i_s_t_ics** | characteristics | characteristics |
| | 3 | **characteristic_s** | characteristics | characteristics |
| | 4 | **cha_rac_t_e_r_istic_s** | characteristics | characteristics |
| | 5 | **ch_ar_act_e_r_istic_s** | characteristics | characteristics |
| 0.5 | 1 | characteristics | characteristics | **characteristic_s** |
| | 2 | characteristics | **c_har_ac_ter_istics** | characteristics |
| | 3 | characteristics | characteristics | **char_acter_istics** |
| | 4 | characteristics | **char_ac_ter_istics** | characteristics |
| | 5 | **characteristic_s** | **character_ist_ics** | characteristics |
| 0.9 | 1 | characteristics | **c_h_a_r_a_c_t_er_i_s_t_i_c_s** | **char_a_c_t_e_ri_s_t_i_c_s** |
| | 2 | characteristics | **char_ac_t_er_ist_ics** | **c_har_a_c_t_e_r_istics** |
| | 3 | characteristics | **c_h_ar_a_c_t_er_i_s_t_ic_s** | **ch_a_r_acter_i_s_t_i_c_s** |
| | 4 | characteristics | **c_h_a_r_ac_t_e_r_i_s_ti_c_s** | **character_i_s_t_i_cs** |
| | 5 | characteristics | **c_ha_ra_ct_er_i_st_i_c_s** | **character_i_stic_s** |

Table 12: Examples of tokenized words using three methods with different hyperparameters for five trials. "_" indicates token boundaries. The vocabularies for each method were constructed using the APG dataset. Sampled tokenizations that differed from the original tokenizations without subword regularization are indicated in bold. We removed special symbols indicating the beginning or middle of words such as "##" for simple explanation.