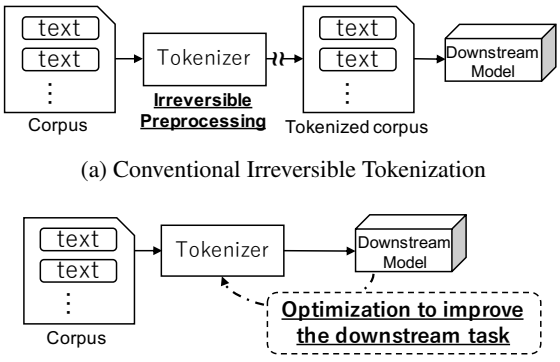
# Optimizing Word Segmentation for Downstream Task

**Tatsuya Hiraoka**[†], **Sho Takase**[†], **Kei Uchiumi**[‡], **Atsushi Keyaki**[‡], **Naoaki Okazaki**[†]

[†] Tokyo Institute of Technology
[‡] Denso IT Laboratory, Inc.
{tatsuya.hiraoka, sho.takase}@nlp.c.titech.ac.jp
{kuchiumi, akeyaki}@d-itlab.co.jp
okazaki@c.titech.ac.jp

## Abstract

In traditional NLP, we tokenize a given sentence as a preprocessing, and thus the tokenization is unrelated to a target downstream task. To address this issue, we propose a novel method to explore a tokenization which is appropriate for the downstream task. Our proposed method, **op**timizing **tok**enization (OpTok), is trained to assign a high probability to such appropriate tokenization based on the downstream task loss. OpTok can be used for any downstream task which uses a vector representation of a sentence such as text classification. Experimental results demonstrate that OpTok improves the performance of sentiment analysis and textual entailment. In addition, we introduce OpTok into BERT, the state-of-the-art contextualized embeddings and report a positive effect.




(a) Conventional Irreversible Tokenization

(b) Optimizing Tokenization (proposed)

Figure 1: Overview of (a) conventional tokenization and (b) optimizing tokenization proposed herein. We directly optimize the tokenizer to improve the performance of the model for a downstream task using the loss of the target task.

## 1 Introduction

Tokenization is a fundamental problem in natural language processing (NLP). We must split a given sequence into a sequence of words for languages that do not contain obvious boundaries, such as Chinese and Japanese. In addition, it is also better to explore appropriate segmentations for languages containing obvious boundaries indicated by whitespaces, such as English (Peng and Dredze, 2015, 2016; Sennrich et al., 2016; He and Sun, 2017; A and Augenstein, 2020; Bollegala et al., 2020).

In traditional NLP, we tokenize a given sentence as a preprocessing. Thus, as shown in Figure 1(a), we apply an existing tokenizer to the given sentence, and then input the tokenized sentence into a model for a target downstream task. In the conventional approach, we obtain the most plausible tokenized sentence based on the tokenizer; however, some studies have varied the tokenization using a sampling during the training to enable the downstream model to adapt to various tokenizations (Kudo, 2018; Hiraoka et al., 2019; Provilkov et al., 2019). Although such a strategy makes the downstream model robust, little attention has been paid to optimizing the tokenizers for a downstream task. Thus, if we acquire an appropriate tokenization to a downstream task, we might improve the task performance.

By contrast, some studies have used multiple tokenized sentences to prevent the damage depending on the tokenization (Chen et al., 2017; Zhang and Yang, 2018; Yang et al., 2018). Their methods compute various tokenizations for a given sentence, and then encode the tokenizations using an architecture based on the LSTM (Hochreiter and Schmidhuber, 1997). Although their methods prevent the error propagation from the tokenizer, they are intractable when handling all possible tokenizations owing to the computational costs required.

This paper describes an exploration into an appropriate tokenization to the downstream tasks. We propose a novel method to optimize a tokenizer based on the downstream task, as shown in Figure



*Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1341–1351
November 16 - 20, 2020. 

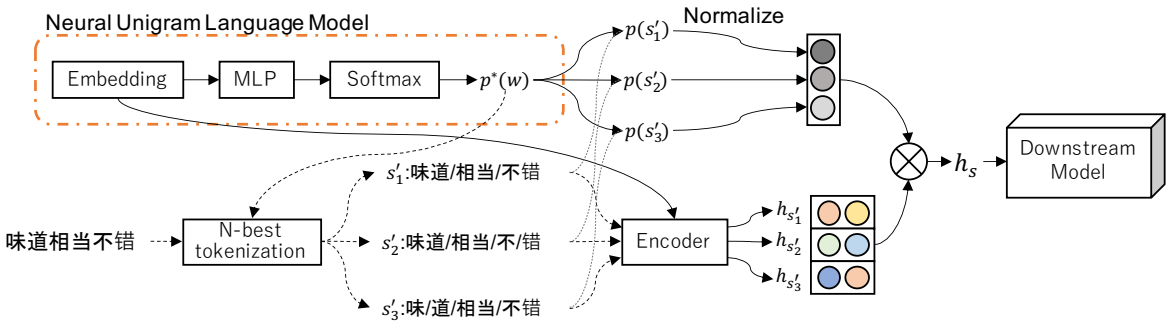



Figure 2: Outline of the proposed method for calculating a sentence vector $\boldsymbol{h}_s$ with the 3-best tokenizations during the training phase. At the inference, we use the 1-best tokenization as well as general neural architectures. The arrows along the continuous line indicate the differentiable paths for backpropagation. We can use various architectures as the *Encoder*, which converts a sequence of tokens into a single vector. *Downstream Model* is the architecture for the downstream tasks, i.e., MLP for text classification.

1(b)[1]. The proposed method generates multiple tokenized sentences as candidates and inputs them into the downstream model. We then update the parameters of the tokenizer to decrease the training loss, and the tokenizer should therefore output a better tokenization for the downstream task.

We design the proposed method to be used for any downstream task that uses a vector representation of a sentence. We conduct experiments on text classification in three languages, and show the effectiveness of the proposed method. Moreover, we indicate that we can also introduce our proposed method into the pre-trained architecture. We combine the proposed method with the state-of-the-art contextualized embeddings, BERT (Devlin et al., 2018), and improve its performance.

## 2 Proposed Method: OpTok

### 2.1 Model Outline

We propose a new architecture for **op**timizing **tok**enization, OpTok. OpTok explores an appropriate tokenization for a downstream task. In other words, OpTok explores a tokenization that yields a better score for a downstream task. Formally, OpTok converts a given sentence $s$ into a sequence of tokens in vocabulary $w \in V$, i.e., $s' = w_1 ... w_i ... w_I$, where $I$ is the number of tokens included in the sentence. In addition, the downstream model achieves the best score with $s'$ among all possible tokenized sentences. Thus, let $q(\cdot)$ be an evaluation function, $z$ be a ground truth of the downstream task, and $f(\cdot)$ be the downstream model, i.e., any neural architecture, and we search the tokenization $s'$ that maximizes the score of the downstream task: $\mathrm{argmax}_{s'}(q(z, f(s')))$.

To search $s'$ satisfying $\mathrm{argmax}_{s'}(q(z, f(s')))$, we train OpTok based on the score of the downstream task $(q(z, f(s')))$. Thus, we optimize both OpTok and the downstream model simultaneously in contrast to a traditional pipeline approach, which tokenizes a given sentence as a preprocessing. OpTok generates multiple tokenized sentences as candidates, and we train OpTok to assign a high probability to a better tokenization based on the score of the downstream task. During the inference step, we make OpTok output only the most plausible tokenized sentence to reduce the computational costs.

Figure 2 shows an overview of OpTok with the downstream model during training. OpTok constructs $N$ tokenized sentences and converts them into vector representations with a neural encoder. Then, OpTok combines the probabilities of each tokenization with the vector representations. We compute the sum of the vector representations weighted by the probabilities, and then input it into the downstream model. Thus, OpTok becomes to assign a high probability to the tokenization which improves the performance of the downstream task. We therefore can obtain $s'$ satisfying $\mathrm{argmax}_{s'}(q(z, f(s')))$ through the training. We describe the details of each module in this section.

### 2.2 Neural Unigram Language Model

OpTok calculates the probability of a token $p(w)$ with a neural unigram language model as follows:

$$d_w = \mathrm{MLP}(\boldsymbol{v}_w), \quad (1)$$

$$p(w) = \frac{\exp(d_w)}{\sum_{\hat{w} \in V} \exp(d_{\hat{w}})}, \quad (2)$$

where MLP is a multilayer perceptron containing trainable parameters, and $\boldsymbol{v}_w$ is an embedding of

[1]Code: https://github.com/tatHi/optok

the word $w$.

To stabilize the learning, as explained in Section 2.5, we employ the smoothed distribution of unigram probability (Kudo, 2018) with a hyperparameter $\alpha$. Concretely, we obtain the smoothed probability as $p^*(w) = \frac{p(w)^\alpha}{\sum_{\hat{w} \in V} p(\hat{w})^\alpha}$. We convert a sentence into a sequence of tokens depending on a probability of a tokenized sentence:

$$p(s') = \prod_{w \in s'} p^*(w). \quad (3)$$

We initialize vocabulary $V$ with a reasonable number of tokens. To choose the initial vocabulary, both supervised and unsupervised word segmentation methods are available, e.g., publicly available pre-trained tokenizers (Kudo, 2006; Yang et al., 2017) and vocabulary acquired using unsupervised word segmentation (Goldwater et al., 2006; Mochihashi et al., 2009; Sennrich et al., 2016). In this study, we use SentencePiece (Kudo and Richardson, 2018) for initialization.

### 2.3 Module for Selecting Tokenization

OpTok generates multiple tokenized sentences as candidates and converts them into a single vector using their probabilities during the training phase.

First, we obtain the $N$-best tokenization of the sentence $s'_1, ..., s_n, ..., s'_N$. We obtain the $N$-best tokenization using the Forward-DP Backward-A* algorithm (Nagata, 1994) for the probabilities produced using the language model mentioned in Section 2.2.

Second, we convert the tokenized sequences into the vectors $\boldsymbol{h}_{s'_n}$ severally as follows:

$$\boldsymbol{h}_{s'_n} = g(s'_n), \quad (4)$$

where $g(\cdot)$ is a neural encoder, which encodes the sequence of tokens, such as those using a CNN and BiLSTM. We found that the learning is stabilized by sharing word embeddings between the encoder and the neural unigram language model.

Finally, we calculate the final vector of the sentence by weighting the vectors of the candidates using their probabilities calculated through Eq. (3) as follows:

$$a_n = \frac{p(s'_n)}{\sum_{m=1}^{N} p(s'_m)}, \quad (5)$$

$$\boldsymbol{h}_s = \sum_{n=1}^{N} a_n \boldsymbol{h}_{s'_n}. \quad (6)$$

Similarly to the attention mechanism, we normalize the probability to meet a restriction $\sum_{n=1}^{N} a_n = 1$[2].

We can use such a vector $\boldsymbol{h}_s$ in the same way as the general encoded vectors. For example, we can construct a neural text classifier by converting $\boldsymbol{h}_s$ into a label-sized vector with an MLP. Updating the entire model with the training loss such as the cross-entropy loss against the gold label, the language model becomes to assign the higher probability to the useful tokenization for the downstream task. At the inference, we obtain the optimal tokenization using the Viterbi algorithm (Viterbi, 1967).

### 2.4 Restricting Vocabulary

To mitigate the local optima which uses longer and more unique tokens for each sentence, we introduce a restriction for the size of the vocabulary during training. Concretely, OpTok constructs the restricted vocabulary $V'$ sampled from the original vocabulary $V$, where $|V'| \leq |V|$, at the beginning of each mini-batch and uses $V'$ as the vocabulary in the mini-batch. The sampling is processed based on the smoothed probability of tokens $p^*(w)$, mentioned in Section 2.2. Then, we calculate the new probability distribution of tokens in $V'$ by normalizing probabilities of them. Moreover, OpTok prepares the embeddings for entire tokens in $V$ but we treat a token outside $V'$ as an unknown token. At the inference, we construct vocabulary by taking the top-$|V'|$ tokens from $V$ based on the updated token probabilities obtained by Eq. (2).

Such sampling of the vocabulary results in the diversity of tokenization in the $N$-best candidates during training. Setting the lower $\alpha$ mentioned in Section 2.2, the distribution of the tokens becomes flatter, and the model can sample various tokens for $V'$. In addition, through the sampling process, we can reduce the importance of words that are unuseful in $V$ for the downstream task. This procedure is related to a vocabulary restriction with a continuous cache technique (Grave et al., 2016; Kawakami et al., 2017).

### 2.5 Maintaining Nature of Language Model

Since the optimization of OpTok only depends on the loss function for the downstream task, the language model of OpTok might be much different from the unigram language model (i.e., frequency

[2]We tried an alternative approach to sampling a plausible tokenization using Gumbel softmax (Jang et al., 2016), but found that it causes instability in the learning.

| | Positive | Negative | Neutral | Total |
|---|---|---|---|---|
| Weibo(Zh) | 407,057 | 263,995 | - | 671,052 |
| Twitter(Ja) | 10,319 | 16,035 | 135,830 | 162,184 |
| Twitter(En) | 56,462 | 43,538 | - | 100,000 |

Table 1: Dataset components on sentiment analysis.

of words) obtained from the training corpus. Meanwhile, we have to keep the corpus-based language model in some cases. To address such cases, we can use the following loss for the sentence $s$ to update the language model using neural EM algorithm (Deligne and Bimbot, 1995; Liang and Klein, 2009; Tran et al., 2016):

$$\mathcal{L}_s^{\mathrm{lm}} = -\sum_{n=1}^{N} a_n \sum_{w \in s'_n} \log p^*(w). \quad (7)$$

We then optimize the weighted sum of the downstream task loss and $\mathcal{L}_s^{\mathrm{lm}}$. Consider text classification as an example. We use cross-entropy loss for the ground-truth label of the sentence $\mathcal{L}_s^{\mathrm{cl}}$. Thus, we optimize the following equation:

$$\mathcal{L}_s = \mathcal{L}_s^{\mathrm{cl}} + \mu \mathcal{L}_s^{\mathrm{lm}}, \quad (8)$$

where $\mu$ is the hyperparameter. Note that we set $\mu = 0$ to confirm the effect of the proposed method in this study.

## 3 Experiments

The goal of this study is to improve the performance of downstream tasks by optimizing the tokenization. Therefore, we evaluate OpTok on various text classification tasks to validate its effect.

### 3.1 Dataset

We evaluate OpTok on text classification, in which a model predicts the label from a text as its input. To confirm the effectiveness of our method on various languages, we utilize datasets in a sentiment analysis for Chinese, Japanese, and English. We employed the corpora on the SNS domain because they have many informal expressions, and thus the difference in tokenization has numerous effects on the performance of the text classification. In addition, we also conducted experiments on the dataset whose sentence contains two kinds of labels to investigate whether OpTok finds different tokenization for each label. Furthermore, we used a textual entailment dataset to indicate that our OpTok can be applied to the task providing two sentences as input. We describe the details of these datasets in the following.

**Weibo(Zh)**[3] is the dataset including short Chinese texts on an SNS with two sentiment labels: *positive* or *negative*. Because the available data are already tokenized with a preprocesser, we detokenize them by removing the whitespaces.

**Twitter(Ja)**[4] is a dataset of short Japanese texts from an SNS about products such as electric appliances. The samples of this dataset initially have five sentiment labels for the target topic: *positive*, *negative*, *neutral*, *both positive and negative*, and *unrelated*. As of the summer of 2018, 352,554 tweets were available, and we extracted only tweets with a single sentiment label of *positive*, *negative*, or *neutral*. In other words, we removed *both positive and negative* and *unrelated* to prevent confusion.

**Twitter(En)**[5] is a dataset of short English texts from an SNS with two sentiment labels: *positive* or *negative*. We exploited this corpus without any preprocessing.

**SNLI** (Bowman et al., 2015) is a widely used dataset for recognizing a textual entailment, which is a text classification requiring two input sentences in English. We employed this dataset to validate the performance of OpTok when using multiple sentences. We used the default split of this corpus and only applied the labeled samples following the existing studies.

**Genre&Rating** are datasets in English that we created from Amazon product data[6], which has reviews from 24 product genres, in which each review has an attached rating from a user of 1 to 5. We sampled 5K reviews from each product genre. In this process, we counted the number of tokens in each review based on whitespaces and removed the review which contains more than 200 tokens. We used sampled reviews for rating prediction and genre prediction tasks from the same review texts.

For the sentiment analysis, we randomly split each dataset into a ratio of 8:1:1 for training, validation, and testing. We also split the dataset of the genre and rating prediction into a ratio of 8:1:1 for a well-balanced genre, in which both tasks share the same split. Table 1 shows an overview of each dataset of the sentiment analysis.

[3] https://github.com/wansho/senti-weibo

[4] http://www.db.info.gifu-u.ac.jp/data/Data_5d832973308d57446583ed9f

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

[6] http://jmcauley.ucsd.edu/data/amazon/

| |vocab| | Weibo(Zh) 16K / 32K | Twitter(Ja) 8K / 16K | Twitter(En) 8K / 16K | SNLI 16K / 32K | Genre 16K / 32K | Rating 16K / 32K |
|---|---|---|---|---|---|---|
| SentencePiece | 92.79 | 86.51 | 77.26 | 76.66 | 71.28 | 67.29 |
| SentencePiece x2 | 92.78 | 85.89 | 77.31 | 76.75 | 71.68 | 67.53 |
| OpTok | **92.82** | **86.97** | **78.52** | **77.04** | **71.88** | **67.68** |

Table 2: The results of text classification. We report the averaged F1 scores (%) over five trials. The highest scores are highlighted in bold. |*vocab*| indicates the sizes of vocabularies of *SencencePiece*/*SentencePiece x2* and restricted/initial sizes for OpTok. The number of vocabularies of *SentencePiece* and *SentencePiece x2* are the same as the restricted and initial sizes of OpTok respectively.

### 3.2 Experimental Settings

For the unigram language model in OpTok, we used two-layered perceptron as $\mathrm{MLP}$ in Eq. (1). We used BiLSTM and a linear layer as an encoder to compute $\boldsymbol{h}_{s'}$ in Eq. (4). We applied BiLSTM to the tokenized sentence based on the unigram language model, and then fed max-pooled outputs to the linear layer. In this procedure, we applied activation function $\tanh$ before and after the linear layer. Then, we applied a dropout to the sentence representations with a rate of 0.5. For SNLI, we shared parameters between encoders for the premise and hypothesis and concatenated both encoded representations. As the downstream model, we used three-layered perceptron which outputs a label-sized vector.

We compared our OpTok with SentencePiece (Kudo and Richardson, 2018), which is a widely used tokenizer. Concretely, we obtained a tokenized sentence based on SentencePiece, and then treated the tokenized sentence as an input to the encoder. In other words, we replaced the unigram language model in OpTok with the SentencePiece tokenizer and used one tokenized sentence as an input to the same architecture. Moreover, many studies have reported that training models with a stochastic tokenization lead to a better performance of the downstream tasks than training a model using deterministic tokenization (Kudo, 2018; Hiraoka et al., 2019; Provilkov et al., 2019). Thus, we trained the encoder and downstream model using subword regularization provided by SentencePiece.

We trained the tokenizer model of SentencePiece on the training split of each dataset. We searched the size of the vocabulary among 8K, 16K, 24K, and 32K, and we selected 16K for Twitter(Ja) and Twitter(En), and 32K for Weibo(Zh), SNLI, and Genre&Rating. We also use a vocabulary obtained by SentencePiece as the initial vocabulary of OpTok for each task, and we initialized the neural unigram language model of OpTok by training the probabilities of its tokens to minimize KL divergence loss against the probabilities obtained using SentencePiece.

We then pre-trained the word embeddings with a bidirectional language model task on the training split of each dataset, and fixed them during the training of the text classification. Because the optimal tokenization is unclear during pre-training, we trained the bidirectional language model with sampling tokenization on each training epoch using SentencePiece. For Genre&Rating, we used the same word embeddings pre-trained on the training split. We did not use any outside resources for pre-training other than the training split.

We trained OpTok and the downstream model using a cross-entropy loss for the gold labels. We employed Adam (Kingma and Ba, 2014) to update the parameters with the default settings of PyTorch.

We set the smoothing hyperparameter $\alpha$ as $0.2$ for both SentencePiece and OpTok as encouraged in Kudo (2018). For the training of our method, the size of the $N$-best tokenization of our method is $N = 3$, and the size of the restricted vocabulary $|V'|$ is half of the initial vocabulary size. At the inference, we used the 1-best tokenization and top-$|V'|$ of the vocabulary based on the language model. We conducted the experiments five times from a random initialization, except for the pre-trained parameters, and reported the averaged F1 score in the result. The maximum training epoch was 20, and we selected a model with the highest performance on the validation split and evaluated it on the test split for each trial.

### 3.3 Results

Table 2 shows the performance of the downstream models using OpTok and SentencePiece. For SentencePiece, we report the results when we set the vocabulary size identical to the restricted and initial vocabulary size of OpTok (*SentencePiece* and *SentnecePiece x2* respectively).

The experimental results demonstrate that the proposed method contributes to improving the per-

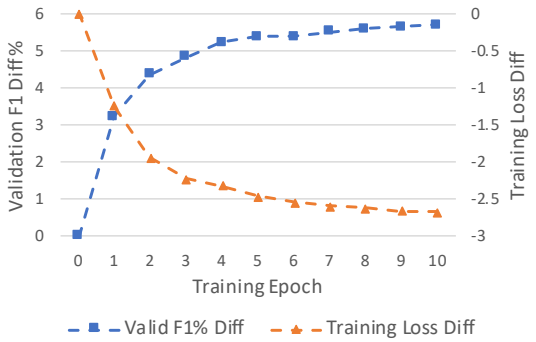



Figure 3: Averaged improvement (difference from the values at the beginning of the training) of validation F1 score and training loss on Twitter(Ja) over five trials only by updating tokenization with OpTok.

formance of the text classification for each language and each task. The performance of OpTok was higher than the method trained using SentencePiece on both sized vocabularies. These results show that OpTok is superior to SentencePiece on the downstream tasks in our experiments.

The results of SNLI show that we can apply OpTok to the task whose input is multiple sentences. Moreover, OpTok has a positive effect on not only informal (sentiment analysis and Genre&Rating in our experiments) but also formal (SNLI) texts.

The proposed method only uses half of the initial vocabulary size at the inference. This fact validates the idea that OpTok contributes to a vocabulary reduction by selecting useful tokens.

## 4 Discussion

### 4.1 Improvement Only by Tokenization

It is still unclear whether the optimized tokenization leads to the improvement described in Section 3.3 because we trained all components simultaneously. Thus, we investigate whether the optimized tokenization contributes to the improvement of the performance on the downstream task. To validate the effect of only tokenization, we trained only the neural unigram language model in OpTok. In other words, we fixed the neural encoder in OpTok and the downstream model with random initialization. We then checked the improvement of the training loss and the F1 score on the validation split by updating only the parameters of the neural unigram language model for tokenization.

We conducted experiments on Twitter(Ja) under the same setting as described in Section 3 and reported the results in Figure 3. Figure 3 shows the difference in the training loss and the validation F1 score from the values at the beginning of the training. This figure indicates the training loss decreases corresponding to the number of epochs, whereas the validation F1 score increases. The results indicate that OpTok explored the tokenization which improved the task performance, and imply that the optimized tokenization contributed to improving the total performance in Section 3.3.

| Genre | | Rating | |
|---|---|---|---|
| Token | Diff | Token | Diff |
| gun | 0.0347 | However | 0.1410 |
| grip | 0.0261 | BUT | 0.1169 |
| zombie | 0.0226 | bad | 0.0532 |
| professional | 0.0190 | paced | 0.0366 |
| treat | 0.0169 | Funk | 0.0299 |
| gray | 0.0148 | awesome | 0.0284 |
| soap | 0.0148 | Ok | 0.0260 |
| dry | 0.0133 | watch | 0.0208 |
| collection | 0.0097 | game | 0.0205 |
| sleeper | 0.0094 | Build | 0.0189 |
| instant | 0.0077 | daughter | 0.0185 |
| phone | 0.0073 | great | 0.0167 |
| tea | 0.0068 | There | 0.0159 |
| scary | 0.0065 | brand | 0.0138 |
| riddled | 0.0063 | what | 0.0122 |

Table 3: Token ranking based on the positive difference in probabilities between the initial and learned language model of OpTok on genre and rating prediction.

### 4.2 Task Oriented Tokenization

We are also interested in whether the optimized tokenization is different from each other when we address the different downstream tasks. To confirm this, we analyzed the results of the Genre&Rating prediction, mentioned in Section 3. The dataset contains two different tasks tied to the same review corpus.

Table 3 shows the ranking of tokens whose probability significantly rise from the initial value on the genre and rating prediction tasks. The optimized neural unigram language model assigned higher scores to tokens that are useful for each task, e.g., *zombie* for Genre and *bad* for Rating. This result demonstrates that OpTok optimizes the tokenization to use helpful tokens frequently. Note that the difference in the probability is vast for the reason mentioned in Section 2.5.

We extracted an example of optimized tokenization from the validation split, which includes the difference in tokenization caused by tasks shown in Table 4. In the tokenization optimized for the genre prediction task, the model cut off an inflection of *book-s* to generalize the token *book* for predicting the proper genre, whereas the model optimized

| Method(true label) | Tokenization |
|---|---|
| SentencePiece | The characters were **interesting** in this book . [...] I will look for additional **books** by Ms . T ate . |
| OpTok (Genre: Book) | The characters were **interesting** in this book . [...] I will look for additional **book s** by Ms . Ta te . |
| OpTok (Rating: 4) | The characters were **interest ing** in this book . [...] I will look for additional **books** by Ms . T ate . |

Table 4: Tokenization acquired by OpTok for different tasks: genre detection and rating detection of the same text. The gold genre label is *Book*, and the rating is 4. Tokens highlighted in bold shows a remarkable difference among the three methods.

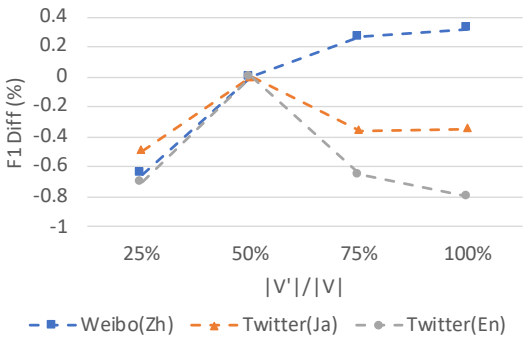



Figure 4: The difference from the score reported in Table 2 against the different $|V'|$ on a sentiment analysis. The full sizes of the vocabulary are 16K for Twitter(Ja) and Twitter(En), and 32K for Weibo(Zh). The size of $N$-best is $N = 3$.

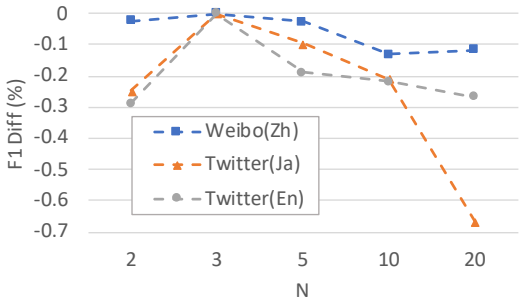



Figure 5: The difference from the score reported in Table 2 against different values of $N$ for a sentiment analysis. The ratio of the restricted vocabulary to the initial vocabulary is 50%.

for rating prediction separated the derivation of *interest-ing* to recognize similar tokens such as *interested* and *interests* in the same manner as a useful token for rating detection. In addition, the model for genre prediction does not split *interest-ing*, and the model for rating detection does not split *books*. This example shows that OpTok can optimize the tokenization for the downstream task.

### 4.3 Effect of Hyperparameter

In this paper, we introduce two hyperparameters to control OpTok: the size of restricted vocabulary and $N$ for $N$-best tokenization. We report the effect of the hyperparameters on the performance of sentiment analysis.

Figure 4 reports the effect caused by the size of the restricted vocabulary in each language. We checked the performance achieved by our method, for which we decreased the size of the vocabulary to 25%, 50% (the default settings used in other experiments), 75%, and 100% of the initial size. In the figure, we show a difference in the averaged F1 scores over five trials from scores reported in Table 2. As shown in the figure, restricting the vocabulary size to 50% contributes to an improved performance of Japanese and English datasets. These results validate that the restriction of the vocabulary works well for the proposed method. Meanwhile, decreasing the size of the vocabulary negatively impacts the performance proportionately to the Chinese dataset. In fact, the average best performance achieved by the full size of the vocabulary, 100% for 32K, was 93.14, which is higher than the score of OpTok shown in Table 2 by 0.32%. This result suggests that decreasing the size of the vocabulary is unnecessary for languages holding vast types of characters because such a restriction leads to a leaking of useful tokens and the production of many unknown tokens in both the training and evaluation, as reported by Hiraoka et al. (2019).

Figure 5 shows the effect of $N$ on the performance of a sentiment analysis. For all languages, $N = 3$ achieves the best performance, whereas increasing $N$ decreases the performance. We consider the reason for the decline to be the gap in the encoding strategies between the training and evaluation. By using a larger $N$, a task-specific module, such as MLP for text classification, is trained using the weighted-sum of the various tokenizations, whereas the module takes a sentence representation encoded with the best tokenization in the inference.

### 4.4 Application for BERT

Numerous studies have recently been focused on exploiting pre-trained language models to enhance the NLP tasks such as BERT (Devlin et al., 2018). In this subsection, we demonstrate that OpTok is applicable to recent NLP modules based on BERT

| Method | F1% |
|---|---|
| Best w/o $\mathrm{BERT}_{\mathrm{base}}$ (from Table 2) | 78.52 |
| $\mathrm{BERT}_{\mathrm{base}}$ | 81.68 |
| + Sampling tokenization | 81.51 |
| + OpTok | **82.03** |

Table 5: F1 scores on sentiment analysis of Twitter(En) with $\mathrm{BERT}_{\mathrm{base}}$. The values highlighted in bold are the highest scores.

by an experiment on Twitter(En).

We replaced the BiLSTM encoder with BERT and conducted the same experiments as mentioned in Section 3. We employed $\mathrm{BERT}_{\mathrm{base}}$ from HuggingFace[7] and fine-tuned its parameters except those of the word embeddings as well as the above experiments. Because the distributed tokenizer for $\mathrm{BERT}_{\mathrm{base}}$ is based on WordPiece, which does not include the probabilities for each piece, we estimate the probabilities on the training split using the EM algorithm (Deligne and Bimbot, 1995; Liang and Klein, 2009; Kudo and Richardson, 2018) and initialized the language model of our method with these probabilities. We did not use a restricted vocabulary because the vocabulary of BERT contains many tokens unrelated to our experiment. Compared to the vocabulary initialized using SentencePiece on only the training split, restricting the vocabulary results in too little diversity of the $N$-best tokenization to cause overfitting of tokenization. We therefore found that it is not necessary to restrict the vocabulary, similar to the Chinese dataset mentioned in Section 4.3. We fine-tuned the parameters of $\mathrm{BERT}_{\mathrm{base}}$ using AdamW (Loshchilov and Hutter, 2017) while updating the neural unigram language model in OpTok with Adam.

Table 5 shows the results of this experiment. We tokenized the corpus using the longest-first matching of WordPiece of $\mathrm{BERT}_{\mathrm{base}}$. We trained the model of +*Sampling tokenization* with a stochastic tokenization like SentencePiece based on the language model initialized using the EM algorithm.

The results show that the pre-trained BERT improved the performance when comparing the scores to those in Table 2. In addition, incorporating OpTok with the BERT beat the original BERT system as well as the method using sampling tokenization. This result indicates that OpTok contributes to an improvement in the popular NLP architecture in terms of optimizing the tokenization.

[7] https://github.com/huggingface/transformers

## 5 Related Work

Numerous studies have aimed to improve the NLP tasks from the perspective of tokenization. Some studies have proposed an approach to prevent segmentation errors by encoding multiple tokenizations jointly. Recent studies investigated Lattice LSTM, which expands LSTM to allow multiple segmentations to be taken as a lattice (Chen et al., 2017; Zhang and Yang, 2018; Yang et al., 2018). Li et al. (2020) followed them to utilize a transformer.

Subword regularization is a famous solution to this problem (Kudo, 2018; Kudo and Richardson, 2018). The authors demonstrated that training models with various tokenizations contribute to an improved performance for machine translations. Provilkov et al. (2019) followed this approach by dropping tokens during the BPE process to vary tokenization, and Hiraoka et al. (2019) by updating the language model during training.

Optimization of the tokenization has attracted attention mainly in the field of machine translation. Some studies have attempted to optimize the tokenization using simple criteria for a machine translation (Xu et al., 2008; Chung and Gildea, 2009; Nguyen et al., 2010; Mermer et al., 2013). Recent studies also tackled this issue for generation tasks. Salesky et al. (2020) developed Incremental BPE, which automatically defines the number of BPE's merge operation for neural machine translation. He et al. (2020) proposed a neural architecture to find a better subword sequence from both the source and target corpora by enhancing the study in Chan et al. (2016). Our work differs from the existing research in that the proposed method is appliable to various neural encoders, and we can optimize the tokenization directly using only backpropagation from the training loss of the downstream tasks without any hand-crafted criteria.

## 6 Conclusion

In this paper, we propose OpTok which explores an appropriate tokenization to the target downstream task. We combine OpTok with the downstream model and train them simultaneously. The experimental results indicate that OpTok improves the performance of several downstream tasks through better tokenization. Moreover, OpTok also has a positive effect on pre-trained contextualized word embeddings such as BERT.

## Acknowledgments

These research results were obtained from the commissioned research by National Institute of Information and Communications Technology (NICT), Japan.

## References

Pranav A and Isabelle Augenstein. 2020. 2kenize: Tying subword sequences for chinese script conversion. *arXiv preprint arXiv:2005.03375*.

Danushka Bollegala, Ryuichi Kiryo, Kosuke Tsujino, and Haruki Yukawa. 2020. Language-independent tokenisation rivals language-specific tokenisation for word similarity prediction. *arXiv preprint arXiv:2002.11004*.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

William Chan, Yu Zhang, Quoc Le, and Navdeep Jaitly. 2016. Latent sequence decompositions. *arXiv preprint arXiv:1610.03035*.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Dag-based long short-term memory for neural word segmentation. *arXiv preprint arXiv:1707.00248*.

Tagyoung Chung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 718–726. Association for Computational Linguistics.

Sabine Deligne and Frederic Bimbot. 1995. Language modeling by variable length sequences: Theoretical formulation and evaluation of multigrams. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 169–172. IEEE.

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*.

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 673–680. Association for Computational Linguistics.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

Hangfeng He and Xu Sun. 2017. F-score driven max margin neural network for named entity recognition in chinese social media. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 713–718.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic programming encoding for subword segmentation in neural machine translation. *arXiv preprint arXiv:2005.06606*.

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.

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

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2017. Learning to create and reuse words in open-vocabulary neural language modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1492–1502.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Taku Kudo. 2006. Mecab: Yet another part-of-speech and morphological analyzer. http://taku910.github.io/mecab/.

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.

Xiaonan Li, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. Flat: Chinese ner using flat-lattice transformer. *arXiv preprint arXiv:2004.11795*.

Percy Liang and Dan Klein. 2009. Online em for unsupervised models. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 611–619.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Coşkun Mermer, Murat Saraçlar, and Ruhi Sarikaya. 2013. Improving statistical machine translation using bayesian word alignment and gibbs sampling. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1090–1101.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 100–108. Association for Computational Linguistics.

Masaaki Nagata. 1994. A stochastic japanese morphological analyzer using a forward-dp backward-a* n-best search algorithm. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 201–207. Association for Computational Linguistics.

ThuyLinh Nguyen, Stephan Vogel, and Noah A Smith. 2010. Nonparametric word segmentation for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 815–823. Association for Computational Linguistics.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 149–155.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2019. Bpe-dropout: Simple and effective subword regularization. *arXiv preprint arXiv:1910.13267*.

Elizabeth Salesky, Andrew Runge, Alex Coda, Jan Niehues, and Graham Neubig. 2020. Optimizing segmentation granularity for neural machine translation. *Machine Translation*, pages 1–19.

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.

Ke M Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Jia Xu, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised chinese word segmentation for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1017–1024. Association for Computational Linguistics.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849.

Jie Yang, Yue Zhang, and Shuailong Liang. 2018. Subword encoding in lattice lstm for chinese word segmentation. *arXiv preprint arXiv:1810.12594*.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564.

| | Rating | | | | | |
|---|---|---|---|---|---|---|
| Genre | 1 | 2 | 3 | 4 | 5 | Total |
| Musical Instruments | 109 | 122 | 352 | 991 | 3,426 | 5,000 |
| Pet Supplies | 254 | 280 | 502 | 848 | 3,116 | 5,000 |
| Video Games | 359 | 234 | 526 | 1,044 | 2,837 | 5,000 |
| CDs and Vinyl | 241 | 221 | 411 | 1,011 | 3,116 | 5,000 |
| Toys and Games | 166 | 156 | 474 | 1,063 | 3,141 | 5,000 |
| Sports and Outdoors | 159 | 181 | 347 | 1,053 | 3,260 | 5,000 |
| Health and Personal Care | 241 | 233 | 479 | 956 | 3,091 | 5,000 |
| Office Products | 97 | 151 | 456 | 1,353 | 2,943 | 5,000 |
| Books | 191 | 238 | 519 | 1,159 | 2,893 | 5,000 |
| Beauty | 272 | 289 | 523 | 996 | 2,920 | 5,000 |
| Baby | 238 | 296 | 513 | 995 | 2,958 | 5,000 |
| Electronics | 333 | 216 | 411 | 943 | 3,097 | 5,000 |
| Patio Lawn and Garden | 198 | 234 | 588 | 1,190 | 2,790 | 5,000 |
| Automotive | 124 | 146 | 354 | 943 | 3,433 | 5,000 |
| Cell Phones and Accessories | 331 | 286 | 564 | 1,000 | 2,819 | 5,000 |
| Grocery and Gourmet Food | 195 | 252 | 567 | 1,000 | 2,986 | 5,000 |
| Clothing Shoes and Jewelry | 206 | 296 | 483 | 1,068 | 2,947 | 5,000 |
| Tools and Home Improvement | 194 | 151 | 387 | 986 | 3,282 | 5,000 |
| Kindle Store | 133 | 143 | 427 | 1,241 | 3,056 | 5,000 |
| Apps for Android | 548 | 277 | 582 | 1,033 | 2,560 | 5,000 |
| Home and Kitchen | 224 | 218 | 338 | 936 | 3,284 | 5,000 |
| Digital Music | 251 | 266 | 498 | 1,181 | 2,804 | 5,000 |
| Amazon Instant Video | 239 | 232 | 530 | 1,121 | 2,878 | 5,000 |
| Movies and TV | 299 | 280 | 526 | 998 | 2,897 | 5,000 |
| Total | 5,602 | 5,398 | 11,357 | 25,109 | 72,534 | 120,000 |

Table 6: Dataset components of Genre&Rating created from Amazon product data.

## A Detailed Experimental Settings

We describe the detailed settings for OpTok used in this study. The size of the word embedding was 64 and the hidden size of BiLSTM was 256. We set the hidden sizes of MLP for the unigram language model and for the downstream tasks as 96 and 256, respectively. The batch size was 256 and the maximum training epoch was 20. We did not search hyperparameters of neural architectures because both OpTok and the baseline system used the same configuration. We described the tuning of the model-specific hyperparameters in Section 4.3. For the experiments using BERT, we set the batch size as 8 due to a memory restriction. For estimating the probabilities of words for the experiments with BERT, the number of iteration of the EM algorithm was 10.

We implemented OpTok with PyTorch. To calculate F1 score, we employed the scikit-learn package. We ran all experiments on a single GPU of NVIDIA Tesla V100 (16GiB).

In Section 3, we created datasets from Amazon product data for a text classification. Table 6 shows the detailed components of the corpus sampled in the way mentioned in Section 3.1. We created both the genre and rating prediction tasks from this corpus.