Article

A Compression-Based Multiple Subword Segmentation for Neural Machine Translation

Keita Nonaka, Kazutaka Yamanouchi, Tomohiro I, Tsuyoshi Okita, Kazutaka Shimada and Hiroshi Sakamoto *

Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-shi, Fukuoka 820-8502, Japan; nonaka.keita331@mail.kyutech.jp (K.N.); yamanouchi.kazutaka437@mail.kyutech.jp (K.Y.); tomohiro@ai.kyutech.ac.jp (T.I.); tsuyoshi@ai.kyutech.ac.jp (T.O.); shimada@ai.kyutech.ac.jp (K.S.)
* Correspondence: hiroshi@ai.kyutech.ac.jp

Abstract: In this study, we propose a simple and effective preprocessing method for subword segmentation based on a data compression algorithm. Compression-based subword segmentation has recently attracted significant attention as a preprocessing method for training data in neural machine translation. Among them, BPE/BPE-dropout is one of the fastest and most effective methods compared to conventional approaches; however, compression-based approaches have a drawback in that generating multiple segmentations is difficult due to the determinism. To overcome this difficulty, we focus on a stochastic string algorithm, called locally consistent parsing (LCP), that has been applied to achieve optimum compression. Employing the stochastic parsing mechanism of LCP, we propose LCP-dropout for multiple subword segmentation that improves BPE/BPE-dropout, and we show that it outperforms various baselines in learning from especially small training data.

Keywords: byte-pair encoding; locally consistent parsing; vocabulary; word embedding

Citation: Nonaka, K.; Yamanouchi, K.; I, T.; Okita, T.; Shimada, K.; Sakamoto, H. A Compression-Based Multiple Subword Segmentation for Neural Machine Translation. *Electronics* **2022**, *11*, 1014. https://doi.org/10.3390/electronics11071014

Academic Editor: Manohar Das

Received: 28 February 2022
Accepted: 21 March 2022
Published: 24 March 2022

1. Introduction

1.1. Motivation

Subword segmentation has been established as a standard preprocessing method in neural machine translation (NMT) [1,2]. In particular, byte-pair encoding (BPE)/BPE-dropout [3,4] is the most successful compression-based subword segmentation. We propose another compression-based algorithm, denoted by LCP-dropout, that generates multiple subword segmentations for the same input; thus, enabling data augmentation especially for small training data.

In NMT, a set of training data is given to the learning algorithm, where training data are pairs of sentences from the source and target languages. The learning algorithm first transforms each given sentence into a sequence of *tokens*. In many cases, the tokens correspond to words in the unigram language model.

The extracted words are projected from a high-dimensional space consisting of all words to a low-dimensional vector space by *word embedding* [5], which enables us to easily handle distances and relationships between words and phrases. The word embedding has been shown to boost the performance of various tasks [6,7] in natural language processing. The space of word embedding is defined by a dictionary constructed from the training data, where each component of the dictionary is called *vocabulary*. Embedding a word means representing it by a set of related vocabularies.

Constructing an appropriate dictionary is one of the most important tasks in this study. Here, consider the simplest strategy that uses the words themselves in the training data as the vocabularies. If a word does not exist in the current dictionary, it is called an unknown word, and the algorithm decides whether or not to register it in the dictionary. Using a sufficiently large dictionary can reduce the number of unknown words as much as desired; however, as a trade-off, overtraining is likely to occur, so the number of vocabularies is usually limited to 16k and 32k. As a result, subword segmentation has been widely used to construct a small dictionary with high generalization performance [8–12].

1.2. Related Works

Subword segmentation is a recursive decomposition of a word into substrings. For example, let the word ‘study’ be registered as a current vocabulary. By embedding other words ’studied’ and ’studying’, we can learn that these three words are similar; however, each time a new word appears, the number of vocabularies grows monotonically.

On the other hand, when we focus on the common substrings of these words, we can obtain a decomposition, such as ‘stud_y’, ‘stud_ied’, and ‘stud_ying’ with the explicit blank symbol ‘_’; therefore, the idea of subword segmentation is not to register the word itself as a vocabulary but to register its subwords. In this case, ‘study’ and ‘studied’ are regarded as known words because they can be represented by combining subwords already registered. These subwords can also be reused as parts of other words (e.g., student and studied), which can suppress the growth of vocabulary size.

In the last decade, various approaches have been proposed along this line. SentencePiece [13] is a pioneering study based on likelihood estimation over the unigram language model, which has high performance. Since maximum likelihood estimation requires quadratic time in the size of training data and the length of the longest subword, a simpler subword segmentation [3] based on BPE [14,15], which is known as one of fastest data compression algorithms, and therefore has many applications, especially in information retrieval [16,17] has been proposed.

BPE-based segmentation starts from a state where a sentence is regarded as a sequence of vocabularies where the set of vocabularies is initially identical to the set of alphabet symbols (e.g., ASCII characters). BPE calculates the frequency of any bigram, merges all occurrences of the most frequent bigram, and registers the bigram as a new vocabulary. This process is repeated until the number of vocabularies reaches the limit. Thanks to the simplicity of the frequency-based subword segmentation, BPE runs in linear time in the size of input string.

However, frequency-based approaches may generate inconsistent subwords for the same substring occurrences. For example, ‘impossible’ and its substring ‘possible’ are possibly decomposed into undesirable subwords, such as ‘po_ss_ib_le’ and ‘i_mp_os_si_bl_e’, depending on the frequency of bigrams. Such merging disagreements can also be caused by misspellings of words or grammatical errors. BPE-dropout [4] proposed a robust subword segmentation for this problem by ignoring each merge with a certain probability. It has been confirmed that BPE-dropout can be trained with higher accuracy than the original BPE and SentencePiece on various languages.

1.3. Our Contribution

We propose LCP-dropout: a novel compression-based subword segmentation employing the stochastic compression algorithm, called locally consistent parsing (LCP) [18,19], to improve the shortcomings of BPE. Here, we describe an outline of the original LCP. Suppose we are given an input string and a set of vocabularies, where similarly to BPE, the set of vocabularies is initially identical to the set of symbols appearing in the string. LCP randomly assigns the binary label for each vocabulary. Then, we obtain a binary string corresponding to the input string where the bigram ‘10’ works as a landmark. LCP merges any bigram in the input string corresponding to a landmark in the binary string, and adds the bigram to the set of vocabularies. The above process is repeated until the number of vocabularies reaches the limit.

By this random assignment, it is expected that any sufficiently long substring contains a landmark. Furthermore, we note that two different landmarks never overlap each other; therefore, LCP can merge bigrams appropriately, avoiding the undesirable subword segmentation that occurs in BPE. Using these characteristics, LCP has been theoretically shown to achieve almost optimal compression [19]. The mechanism of LCP has also been mainly applied to information retrieval [18,20,21].

A notable feature of the stochastic algorithm is that LCP assigns a new label to each vocabulary for each execution. Owing to this randomness, the LCP-based subword segmentation is expected to generate different subword sequences representing a same input; thus, it is more robust than BPE/BPE-dropout. Moreover, these multiple subword sequences can be considered as data augmentation for small training data in NMT.

LCP-dropout consists of two strategies: landmark by random labeling for all vocabularies and dropout of merging bigrams depending on the rank in the frequency table. Our algorithm requires no segmentation training in addition to counting by BPE and labeling by LCP and uses standard BPE/LCP in test time; therefore, our algorithm is simple. With various language corpora including small datasets, we show that LCP-dropout outperforms the baseline algorithms: BPE/BPE-dropout/SentencePiece.

2. Background

We use the following notations throughout this paper. Let $\mathcal{A}$ be the set of alphabet symbols, including the blank symbol. A sequence S formed by symbols is called a string. $S[i]$ and $S[i,j]$ are i-th symbol and substring from $S[i]$ to $S[j]$ of S, respectively. We assume the meta symbol '$-$' not in $\mathcal{A}$ to explicitly represent each subwords in S. For a string S from $\mathcal{A} \cup \{-\}$, a maximal substring of S including no $-$ is called a subword. For example, $S = a - b - a - a - b / a - b - a - ab$ contains the subwords in $\{a, b\} / \{a, b, ab\}$, respectively.

In subword segmentation, the algorithm decomposes all the symbols in S by the meta symbol. When a trigram $a - b$ is merged, the meta symbol is erased and the new subword ab is added to the vocabulary, i.e., ab is treated as a single vocabulary.

In the following, we describe previously proposed subword segmentation algorithms, called SentencePiece (Kudo [13]), BPE (Sennrich et al. [3]), and BPE-dropout (Provilkov et al. [4]). We assume that our task in NMT is to predict a target sentence T given a source sentence S, where these methods including our approach are not task-specific.

2.1. SentencePiece

SentencePiece [13] can generate different segmentations for each execution. Here, we outline SentencePiece in the unigram language model. Given a set of vocabularies, V, a sentence T, and the probability $p(x)$ of occurrence of $x \in V$, the probability of the partition $x = (x_1, \dots, x_n)$ for $T = x_1 \cdots x_n$ is represented as $P(x) = \Pi_{i=1}^{n} p(x_i),\ x_i \in V$, where $\Sigma_{x \in V} p(x) = 1$. The optimum partition x^* for T is obtained by searching for the x that maximizes $P(x)$ from all candidate partitions $x \in S(T)$.

Given a set of sentences, D, as training data for a language, the subword segmentation for D can be obtained through the maximum likelihood estimation of the following $\mathcal{L}$ with $P(x)$ as a hidden variable by using EM algorithm, where $X^{(s)}$ is the s-th sentence in D.

$$\mathcal{L} = \sum_{s=1}^{|D|} \log P(X^{(s)}) = \sum_{s=1}^{|D|} \log \left(\sum_{x \in S(X^{(s)})} P(x) \right)$$

SentencePiece was shown to achieve significant improvements over the method based on subword sequences; however, this method is rather complicated because it requires a unigram language model to predict the probability of subword occurrence, EM algorithm to optimize the lexicon, and Viterbi algorithm to create segmentation samples.

2.2. BPE and BPE-Dropout

BPE [14] is one of practical implementations of Re-pair [15], which is known as the algorithm with the highest compression ratio. Re-pair counts the frequency of occurrence of all bigrams xy in the input string T. For the most frequent xy, it replaces all occurrences of xy in T such that $T[i, i+1] = xy$, with some unused character z. This process is repeated until there are no more frequent bigrams in T. The compressed T can be recursively decoded by the stored substitution rules $z \to xy$.

Since the naive implementation of Re-pair requires $O(|T|^2)$ time, we use a complex data structure to achieve linear time; however, it is not practical for large-scale data because it consumes $\Omega(|T|)$ of space. As a result, we usually split $T = t_1 t_2 \cdots t_m$ into substrings of a constant length and process each t_i by the naive Re-pair without special data structure, called BPE. Naturally, there is a trade-off between the size of the split and the compression ratio. BPE-based subword segmentation [3] (called BPE simply) determines the priority of bigrams according to their frequency and adds the merged bigrams as the vocabularies.

Since BPE is a deterministic algorithm, it splits a given T in one way. Thus, it is not easy to generate multiple partitions such as the stochastic approach (e.g., [13]). As a result, BPE-dropout [4], ignoring the merging process with a certain probability, was proposed. In BPE-dropout, for the current T and the most frequent xy, for each occurrence i satisfying $T[i, i+1] = xy$, merging xy is dropped with a certain small probability p (e.g., $p = 0.1$). This mechanism makes BPE-dropout probabilistic and generates a variety of splits. BPE-dropout has been recorded to outperform SentencePiece in various languages. Additionally, BPE-based methods are faster and easier to implement than likelihood-based approaches.

2.3. LCP

Frequency-based compression algorithms (e.g., [14,15]) are known to be not optimum from a theoretical point of view. Optimum compression here means a polynomial-time algorithm that satisfies $|A(T)| = O(|A^*(T)| \cdot \log |T|)$ with the output $A(T)$ of the algorithm for the input T and an optimum solution $A^*(T)$. Note that computing $A^*(T)$ is NP-hard [22].

For example, consider a string $T = \cdots \mathrm{abcdefg} \cdots \mathrm{bcdefg} \cdots$. Assuming the rank of these frequencies: $freq(\mathrm{ab}) > freq(\mathrm{bc}) > freq(\mathrm{cd}) > \cdots$, merging for T is possibly $T = \cdots(\mathrm{ab})(\mathrm{cd})(\mathrm{ef})(\mathrm{g} \cdots (\mathrm{bc})(\mathrm{de})(\mathrm{fg}) \cdots$; however, the desirable merging would be $T = \cdots \mathrm{a}(\mathrm{bc})(\mathrm{de})(\mathrm{fg}) \cdots (\mathrm{bc})(\mathrm{de})(\mathrm{fg}) \cdots$ considering the similarity of these substrings.

Since such pathological merging cannot be prevented by frequency information alone, frequency-based algorithms cannot obtain asymptotically optimum compression [23]. Various linear time and optimal compressions have been proposed to improve this drawback. LCP is one of the simplest optimum compression algorithms. The original LCP, similar to Re-pair, is a deterministic algorithm. Recently, the introduction of probability into LCP [19] has been proposed, and in this study, we focus on the probabilistic variant. The following is a brief description of the probabilistic LCP.

We are given an input string $T = a_1 a_2 \cdots a_n$ of length n and a set of vocabularies, V. Here, V is initialized as the set of all characters appearing in T.

1. Randomly assign a label $L(a) \in \{0, 1\}$ to each $a \in V$.
2. According to $L(a)$, compute the sequence $L(T) = L(a_1)L(a_2) \cdots L(a_n) \in \{0, 1\}^n$.
3. Merge all bigram $a_i a_{i+1}$ provided $L(T)[i, i+1] =$ '10'.
4. Set $V = V \cup \{a_i a_{i+1}\}$ and repeat the above process.

The difference between LCP and BPE is that BPE merges bigrams with respect to frequencies, whereas LCP pays no attention to them. Instead, LCP merges based on the binary labels assigned randomly. The most important point is that any two occurrences of '10' never overlap. For example, when T contains a trigram abc, there is no possible assignment allowing $(ab)c$ and $a(bc)$ simultaneously. By this property, LCP can avoid the problem that frequently occurs in BPE. Although LCP theoretically guarantees almost optimum compression, as far as the authors know, this study is the first result of applying LCP to machine translation.

3. Our Approach: LCP-Dropout

BPE-dropout allows diverse subword segmentation for BPE by ignoring bigram merging with a certain probability; however, since BPE is a deterministic algorithm, it is not trivial to generate various candidates of bigram. In this study, we propose an algorithm that enables multiple subword segmentation for the same input by combining the theory of LCP with the original strategy of BPE.

3.1. Algorithm Description

We define the notations used in our LCP-dropout (Algorithm 1) and its subroutine (Algorithm 2). Let $\mathcal{A}$ be an alphabet and '_' be the explicit blank symbol not in $\mathcal{A}$. A string w formed from $\mathcal{A}$ is called *word*, denoted by $x \in \mathcal{A}^*$, and a string $s \in (\mathcal{A} \cup \{\_\})^*$ is called a *sentence*.

We also assume the meta symbol '$-$' not in $\mathcal{A} \cup \{\_\}$. By this, a sentence x is extended to have all possible merges: Let $\tilde{x}$ be the string of all symbols in x separated by $-$, e.g., $\tilde{x} = a - b - a - b - b$ for $x = ababb$. For strings x and y, if y is obtained by removing some occurrences of $-$ in x, then we express the relation $y \preceq x$ and y is said to be a *subword segmentation* of x.

After merging $a - b$ (i.e., $a - b$ is replaced by ab), the substring ab is treated as a single symbol. Thus, we extend the notion of bigram to vocabularies of length more than two. For a string of the form $s = \alpha_1 - \alpha_2 - \cdots - \alpha_n$ such that each α_i contains no $-$, each $\alpha_i - \alpha_{i+1}$ is defined to be a bigram consisting of the vocabularies α_i and α_{i+1}.

Algorithm 1 LCP-dropout.

```
Input: $\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n\}$ for a set of sentences, $X = \{x_1, x_2, \ldots, x_n\}$, and hyperparameters $(v, \ell, k)$ {$v > 0$: #total vocabularies, $0 < \ell \leq v$: #partial vocabularies, $k \in (0,1]$: threshold of frequencies}
Output: Set of subword sequences, $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_m\}$, where $Y_i = (y_1^{(i)}, y_2^{(i)}, \ldots, y_n^{(i)})$ satisfies $y_j^{(i)} \preceq x_j$, $|V| = |\bigcup_{1 \leq i \leq m} V(Y_i)| \leq v$ and $|V_i| = |V(Y_i)| \leq \ell$
1:  $m \leftarrow 1$ and $Y_m \leftarrow \tilde{X}$
2:  while (TRUE) do
3:    initialize $V_m, FREQ(k)$
4:    while ($|V_m| < \ell$) do
5:      $LCP(Y_m, V_m, FREQ(k))$
6:    end while
7:    if ($|V| \geq v$) then
8:      return $\mathcal{Y} = \{Y_1, \ldots, Y_m\}$
9:    end if
10:   $m \leftarrow m + 1$, $Y_m \leftarrow \tilde{X}$
11: end while
```

Algorithm 2 $LCP(Y, V, FREQ(k))$ %subroutine of LCP-dropout.

```
1: assign $L : V \to \{0,1\}$ randomly
2: $FREQ(k) \leftarrow$ the set of top-$k$ frequent bigrams in $Y$ of the form $\alpha - \beta$ with $L(\alpha\beta) =$ '10'
3: merge all occurrences of $\alpha - \beta$ in $Y$ for each $\alpha - \beta \in FREQ(k)$
4: add all the vocabularies $\alpha\beta$ to $V$
```

3.2. Example Run

Table 1 presents an example of subword segmentation using LCP-dropout. Here, the input X consists of a single sentence $ababcaacabcb$. The hyperparameters are $(v, \ell, k) = (6, 5, 0.5)$. First, the set of vocabularies is initialized to $V = \{a, b, c\}$; for each $\alpha \in V$, a label $L(w) \in \{0,1\}$ is randomly assigned (depth 0). Next, find all occurrences of 10 in L, and the corresponding bigrams are merged depending on their frequencies. Here, $L(ab) = L(ac) = 10$ but only $a - b$ is top-k bigram assigned 10, and then $a - b$ is merged to ab. The resulting string is shown in the depth 1 over the new vocabularies $V_1 = \{a, b, c, ab\}$. This process is repeated while $|V_m| < \ell$ for the next m. The condition $|V_2| = 5$ terminates the inner-loop of LCP-dropout, and then the subword $Y_1 = ab - abc - a - a - c - abc - b$ is generated. Since $|V(Y_1)| < v$, the algorithm generates the next subword segmentations Y_2 for the same input.

Finally, we obtain the multiple subword segmentation $Y_1 = ab - abc - a - a - c - abc - b$ and $Y_2 = ab - ab - ca - a - ca - b - c - b$ for the same input string.

Table 1. Example of multiple subword segmentation using LCP-dropout for the single sentence '$x = ababcaacabcb$' with the hyperparameters $(v, \ell, k) = (6, 5, 0.5)$, where the meta symbol $-$ is omitted, and L is the label of each vocabulary assigned by LCP. The resulting subword segmentation is $\mathcal{Y} = \{Y_1, Y_2\}$.

1-st Trial												
input x: depth 0	a	b	a	b	c	a	a	c	a	b	c	b
L	1	0	1	0	0	1	1	0	1	0	0	0
depth 1	ab		ab		c	a	a	c	ab		c	b
L	1		1		0	1	1	0	1		0	1
Y_1: depth 2	ab		abc			a	a	c	abc			b
2-nd Trial												
same x: depth 0	a	b	a	b	c	a	a	c	a	b	c	b
L	0	1	0	1	1	0	0	1	0	1	1	1
depth 1	a	b	a	b	ca		a	ca		b	c	b
L	1	0	1	0	0		1	0		0	0	0
Y_2: depth 2	ab		ab		ca		a	ca		b	c	b

3.3. Framework of Neural Machine Translation

Figure 1 shows the framework of our transformer-based machine translation model with LCP-dropout. Transformer is the most successful NMT model [24]. The model mainly consists of an encoder and decoder. The encoder converts the input sentence in the source language into a word embedding (Emb$_i$ in Figure 1), taking into account the positional information of the characters. Here, the notion of word is extended to that of subword in this study. The subwords are obtained by our proposed method, LCP-dropout. Next, the correspondences in the input sentence are acquired as attention (Multi-Head Attention). Then, the normalization is performed through a forward propagation network formed by linear transformation, activation by ReLU function, and linear transformation. These processes are performed in N = 6 layers for the decoder.

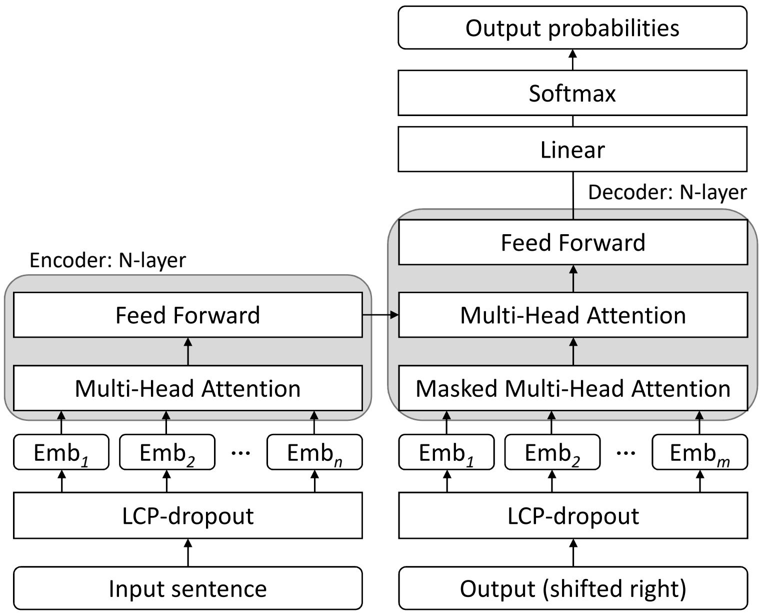

Figure 1. Framework of our neural machine translation model with LCP-dropout.

For the decoder, it receives the candidate sentence generated by the encoder and the input sentence for the decoder. Then, it acquires the correspondence between those sentences as attention (Multi-Head Attention). This process is also performed in N = 6 layers. Finally, the predicted probability of each label is calculated by linear transformation and softmax function.

4. Experimental Setup

4.1. Baseline Algorithms

Baseline algorithms are SentencePiece [13] with the unigram language model and BPE/BPE-dropout [3,4]. SentencePiece takes the hyperparameters l and α, where l specifies how many best segmentations for each word are produced before sampling and α controls the smoothness of the sampling distribution. In our experiment, we used $(l = 64, \alpha = 0.1)$, which performed best on different data in the previous studies.

BPE-dropout takes the hyperparameter p, where during segmentation, at each step, some merges are randomly dropped with the probability p. If $p = 0$, the segmentation is equal to the original BPE and $p = 1$, the algorithm outputs the input string itself. Then, the value of p can be used to control the granularity of segmentation. In our experiment, we used $p = 0$ for the original BPE and $p = 0.1$ for the BPE-dropout with the best performance.

4.2. Data Sets, Preprocessing, and Vocabulary Size

We verified the performance of the proposed algorithm for a wide range of datasets with different sizes and languages. Table 2 summarizes the details of the datasets and hyperparameters. These data are used to compare the performance of LCP-dropout and baselines (SentencePiece/BPE/BPE-dropout) with appropriate hyperparameters and vocabulary sizes shown in [4].

Table 2. Overview of the datasets and hyperparameters. The hyperparameter v (vocabulary size) is common to all algorithms (baselines and ours) and others (ℓ and k) are specific to LCP-dropout only.

Corpus	Language ($L_1 - L_2$)	#Sentences (train/dev/test)	Batch Size	Hyperparameters (v, ℓ, k)
News Commentary v16	$\mathrm{En} - \mathrm{De}$	380k/2808/2906	3072	16k, 16k/8k, 0.01/0.05/0.1
	$\mathrm{En} - \mathrm{Fr}$	357k/3020/3133	3072	16k, 8k, 0.01
	$\mathrm{En} - \mathrm{Zh}$	305k/2968/2936	3072	16k, 8k, 0.01
KFTT	$\mathrm{En} - \mathrm{Ja}$	440k/1166/1160	3072	16k, 8k, 0.01
WMT14	$\mathrm{En} - \mathrm{De}$	4.5M/2737/3004	3072	32k, 32k/16k, 0.01

Before subword segmentation, we preprocess all datasets with the standard Moses toolkit (https://github.com/moses-smt/mosesdecoder, accessed on 5 January 2022) where for Japanese and Chinese, subword segmentations are trained almost from raw sentences because these languages have no explicit word boundaries; thus, Moses tokenizer does not work correctly.

Based on a recent research on the effect of vocabulary size on translation quality, the vocabulary size is modified according to the dataset size in our experiments (Table 2).

To verify the performance of the proposed algorithm for small training data, we use News Commentary v16 (https://data.statmt.org/news-commentary/v16, accessed on 5 January 2022), a subset of WMT14 (https://www.statmt.org/wmt14/translation-task.html, accessed on 1 February 2022), as well as KFTT (http://www.phontron.com/kftt, accessed on 5 January 2022). In addition, we use a large training data in WMT14. The training step is set to 200,000 for all data. In training, pairs of sentences of source and target languages were batched together by approximate length. As shown in Table 2, the batch size was standardized to approximately 3k for all datasets.

4.3. Model, Optimizer, and Evaluation

NMT was realized by the *seq2seq* model, which takes a sentence in the source language as input and outputs a corresponding sentence in the target language [25]. A transformer is an improvement of seq2seq model, that is the most successful NMT [24].

In our experiments, we used OpenNMT-tf [26], a transformer-based NMT, to compare LCP-dropout and other baselines algorithms. The parameters of OpenNMT-tf were set as in the experiment of BPE-dropout [4]. The batch size was set to 3072 for training and 32 for testing. We also used the regularization and optimization procedure as described in BPE-dropout [4].

The quality of machine translation is quantitatively evaluated by BLEU score, i.e., the similarity between the result and the reference of translation. It is calculated using the following formula based on the number of matches in their n-grams. Let t_i and r_i $(1 \leq i \leq m)$ be the i-th translation and reference sentences, respectively.

$$\text{BLEU} = BP_{\text{BLEU}} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), \quad p_n = \frac{\sum_{i=1}^{m} \#n\text{-gram that match in } t_i \text{ and } r_i}{\sum_{i=1}^{m} \#n\text{-gram in } t_i},$$

where N is a small constant (e.g., $N = 4$), and BP_{BLEU} is the brevity penalty when $|t_i| < |r_i|$, where $BP_{\text{BLEU}} = 1$ otherwise. In this study, we use SacreBLEU [27]. For Chinese, we add option`-tok zh` to SacreBLEU. Meanwhile, we use character-based BLEU for Japanese.

5. Experiments and Analysis

All experiments were conducted using the following: OS: Ubuntu 20.04.2 LTS, CPU: Intel(R) Xeon(R) W-2135 CPU @ 3.70 GHz, GPU: GeForce RTX 2080 Ti Rev. A, Memory: 64 GB RAM, Storage: 2TB SSD, Python-3.8.6, SentencePiece-0.1.96 (https://pypi.org/project/sentencepiece/, accessed on 5 January 2022) (Python wrapper for SentencePiece including BPE/BPE-dropout runtime). The numerical results are averages of three independent trials.

5.1. Estimation of Hyperparameters for LCP-Dropout

First, we estimate suitable hyperparameters for LCP-dropout. Table 3 summarizes the effect of hyperparameters (v, ℓ, k) on the proposed LCP-dropout. This table shows the details of multiple subword segmentation using LCP-dropout and BLEU scores for the language pair of English (En) and German (De) from News Commentary v16 (Table 2). For each threshold $k \in \{0.01, 0.05, 0.1\}$, En and De indicate the number of multiple subword sequences generated for the corresponding language, respectively. The last two values are the BLEU scores for De $\rightarrow$ En with $\ell = v$ and $\ell = v/2$ for $v =$ 16k, respectively.

Table 3. Experimental results of LCP-dropout for News Commentary v16 (Table 2) with respect to the specified hyperparameters, where the translation task is De $\rightarrow$ En. Bold indicates the best score.

Top-k Threshold	**#Subword**		**BLEU**	
($k \in (0,1]$)	**En**	**De**	**($\ell = v$)**	**($\ell = v/2$)**
0.01	21.3	7.7	39.0	**39.7**
0.05	4.7	3.7	39.0	**39.4**
0.1	3.3	2.0	38.8	**39.4**

The threshold k controls the dropout rate, and ℓ contributes to the multiplicity of the subword segmentation. The results show that k and ℓ affect the learning accuracy (BLEU). The best result is obtained when $(k, \ell) = (0.01, \ell = v/2)$. This can be explained by the results in Table 4, which show the depth of the executed inner loop of the LCP-dropout for randomly assigning $\{0, 1\}$ to vocabularies, where, when $\ell = v/2$, means the average before the outer-loop terminates. As a result, the larger this value is, the more likely it is that longer subwords will be generated; however, unlike BPE-dropout, the value of k alone is not enough to generate multiple subwords. The proposed LCP-dropout guarantees the diversity by initializing the subword segmentation by ℓ ($\ell < v$). Using this result, we fix $(k, \ell) = (0.01, v/2)$ as the hyperparameter of LCP-dropout.

Table 4. Depth of label assignment in LCP-dropout.

Top-k Threshold ($k \in (0,1]$)	$\ell = v$		$\ell = v/2$	
	En	**De**	**En**	**De**
0.01	83.7	48.0	54.9	35.4
0.05	18.7	12.3	13.0	9.0
0.1	10.3	7.7	7.3	6.0

5.2. Comparison with Baselines

Table 5 summarizes the main results. We show BLEU scores for News Commentary v16 and KFTT: En and De are the same in Table 3. In addition to these languages, we set French (Fr), Japanese (Ja), and Chinese (Zh). For each language, we show the average of the number of multiple subword sequences generated by LCP-dropout. For almost datasets, LCP-dropout outperforms the baselines algorithms. Meanwhile, we use the best ones reported in the previous study for the hyperparameters of BPE-dropout and SentencePiece.

Table 5. Experimental results of LCP-dropout (denoted by LCP), BPE-dropout (denoted by BPE), and SentencePiece (denoted by SP) on various languages in Table 2 (small corpus: News Commentary v16 and KFTT, and large corpus: WMT14), where 'multiplicity' denotes the average number of sequences generated per input string. Bold indicates the best score.

Corpus	**Language (Multiplicity)**	**Translation Direction**	**LCP ($k = 0.01$)**	**BPE ($p = 1$,**	**0.1)**	**SP**
News Commentary v16 (small)	En–De (21.3–7.7)	De → En	**39.7**	35.7	39.1	38.9
		En → De	**28.4**	27.4	27.4	27.5
	En–Fr (23.0–19.3)	Fr → En	**35.1**	34.9	34.9	34.2
		En → Fr	**29.5**	15.2	28.2	28.3
	En–Zh (26.0–8.7)	Zh → En	24.2	24.2	**24.6**	24.2
		En → Zh	**6.5**	2.0	2.1	1.8
KFTT (small)	En–Ja (17.7–10.0)	Ja → En	**20.0**	19.6	19.6	19.2
		En → Ja	**8.5**	3.0	3.6	3.5
WMT14 (large)	En–De (9.3–5.3)	De → En	28.7	28.9	**32.2**	**32.2**

Table 5 extracts the effect of alphabet size on subword segmentation. In general, Japanese (Ja) and Chinese (Zh) alphabets are very large, containing at least 2k alphabet symbols even if we limit them in common use; therefore, the average length of words is small and subword semantics is difficult. For these cases, we confirmed that LCP-dropout has higher BLEU scores than other methods for these languages.

Table 5 also presents the BLEU scores for a large corpus (WMT14) for the translation De → En. This experiment shows that LCP-dropout cannot outperform baselines with the hyperparameter we set. This is because the ratio of the vocabulary size (v, ℓ) to dropout rate k is not appropriate. As data to support this conjecture, it can be confirmed that the multiplicity in the large datasets is much smaller than that of small corpus (Table 5). This is caused by the reduced repetitions of label assignments, as shown in Table 6 compared to Table 4. The results show that the depth of the inner loop is significantly reduced, which is why enough subword sequences cannot be generated.

Table 6. Depth of label assignment for large corpus.

Top-k Threshold ($k \in (0,1]$)	$\ell = v$		$\ell = v/2$	
	En	**De**	**En**	**De**
0.01	24.0	18.0	17.1	14.2

Table 7 presents several translation results. The 'Reference' represents the correct translation for each case, and the BLEU score is obtained from the pair of the reference and each translation result. We also show the average length for each reference sentence indicated by the 'ave./word'. These results show the characteristics of successful and unsuccessful translations by the two algorithms related to the length of words.

Table 7. Examples of translated sentences by LCP-dropout ($k = 0.01$) and BPE-dropout ($p = 0.1$) with the reference translation for News Commentary v16. We show the average word length (ave./word) for each reference sentence as well as the average subword length (ave./subword) generated by the respective algorithms for the entire corpus. We also show the BLEU scores between the references and translated sentences as well as their standard deviations (SD).

Reference: (ave./word = 5.00)	'Even if his victory remains unlikely, Bayrou must now be taken seriously.'	BLEU
LCP-dropout:	'While his victory remains unlikely, Bayrou must now be taken seriously.'	84.5
BPE-dropout:	'Although his victory remains unlikely, he needs to take Bayrou seriously now.'	30.8
Reference: (ave./word = 5.38)	'In addition, companies will be forced to restructure in order to cut costs and increase competitiveness.'	BLEU
LCP-dropout:	'In addition, restructuring will force rms to save costs and boost competitiveness.'	12.4
BPE-dropout:	'In addition, businesses will be forced to restructure in order to save costs and increase competitiveness.'	66.8
ave./subword	4.01 (LCP-dropout): 4.31 (BPE-dropout)	
SD of BLEU	21.98 (LCP-dropout): 21.59 (BPE-dropout)	

Considering subword segmentation as a parsing tree, LCP produces a balanced parsing tree, whereas the tree produced by BPE tends to be longer for a certain path. For example, for a substring $abcd$, LCP tends to generate subwords such as $((ab)(cd))$, while BPE generates them such as $(((ab)c)d)$. In this example, the average length of the former is shorter than that of the latter. This trend is supported by the experimental results in Table 7 showing the average length of all subwords generated by LCP/BPE-dropout for real datasets. Due to this property, when the vocabulary size is fixed, LCP tends not to generate subwords of approximate length because it decomposes a longer word into excessively short subwords.

Figure 2 shows the distributions of sentence length of English. The sentence length denotes the number of tokens in a sentence. BPE-dropout is a well-known fine-grained segmentation approach. The figure shows that LCP-dropout produces more fine-grained segmentation than the other three segmentation approaches; therefore, LCP-dropout is considered to be superior in subword segmentation for languages consisting of short words. Table 5 including the translation results for Japanese and Chinese also supports these characteristics.

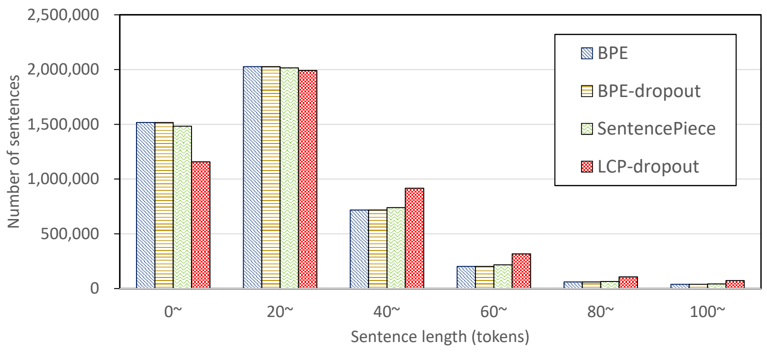

Figure 2. Distribution of sentence length. The number of tokens in each sentence by LCP-dropout tends to be larger than the others: BPE, BPE-dropout, and SentencePiece.

6. Conclusions, Limitations, and Future Research

6.1. Conclusions and Limitations

In this study, we proposed the LCP-dropout as an extension of BPE-dropout [4] for multiple subword segmentation by applying a near-optimum compression algorithm. The proposed LCP-dropout can properly decompose strings without background knowledge of the source/target language by randomly assigning binary labels to vocabularies. This mechanism allows generating consistent multiple segmentations for the same string. As shown in the experimental results, LCP-dropout enables data augmentation for small datasets, where sufficient training data are unavailable on minor languages or limited fields.

Multiple segmentation can also be achieved by likelihood-based methods. After SentencePiece [13], various extensions have been proposed [28,29]. In contrast to these studies, our approach focuses on a simple linear-time compression algorithm. Our algorithm does not require any background knowledge of the language compared to word replacement-based data augmentation, [30,31] where some words in the source/target sentence are swapped with other words preserving grammatical/semantic correctness.

6.2. Future Research

The effectiveness of LCP-dropout was confirmed for almost small corpora. Unfortunately, the optimal hyperparameter obtained in this study did not work well for a large corpus. Further, the learning accuracy was found to be affected by the alphabet size of the language. Future research directions include an adaptive mechanism for determining the hyperparameters depending on training data and alphabet size.

In the experiments in this paper, we considered word-by-word subword decomposition. On the other hand, multi-words are known to violate the compositeness of language; therefore, by considering multi-words as longer words and performing subword decomposition, LCP-dropout can be applied to language processing related to multi-words. In this study, subword segmentation was applied to machine translation. To improve the BLEU score, there are other approaches such as data augmentation [32]. Incorporating the LCP-dropout with them is one interesting approach. In this paper, we handled several benchmark datasets with major languages. Recently, machine translation of low-resource languages is an important task [33]. Applying the LCP-dropout to this task is also important future work.

Although the proposed LCP-dropout is currently applied only to machine translation, we plan to apply our method to other linguistic tasks including sentiment analysis, parsing, and question answering in future studies.

Author Contributions: Conceptualization, H.S.; methodology, T.I.; software, K.N.; validation, T.O. and H.S.; formal analysis, H.S.; investigation, H.S.; resources, K.N.; data curation, K.S. and K.Y.; writing—original draft preparation, H.S.; writing—review and editing, T.O., K.S. and H.S.; visualization, K.S. and H.S.; supervision, H.S.; project administration, H.S.; funding acquisition, H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by JSPS KAKENHI (Grant Number 21H05052, 17H01791).

Acknowledgments: The authors thank all the reviewers who contributed to the reviewing process.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Barrault, L.; Bojar, O.; Costa-jussà, M.R.; Federmann, C.; Fishel, M.; Graham, Y.; Haddow, B.; Huck, M.; Koehn, P.; Malmasi, S.; et al. Findings of the 2019 Conference on Machine Translation (WMT19). In Proceedings of the Fourth Conference on Machine Translation, Florence, Italy, 1–2 August 2019; pp. 1–61.
2. Bojar, O.; Federmann, C.; Fishel, M.; Graham, Y.; Haddow, B.; Koehn, P.; Monz, C. Findings of the 2018 Conference on Machine Translation (WMT18). In Proceedings of the Third Conference on Machine Translation: Shared Task Papers, Belgium, Brussels, 31 October–1 November 2018; pp. 272–303.
3. Sennrich, R.; Haddow, B.; Birch, A. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; pp. 1715–1725.
4. Provilkov, I.; Emelianenko, D.; Voita, E. BPE-Dropout: Simple and Effective Subword Regularization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 1882–1892.
5. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Jeff Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–8 December 2013; pp. 3111–3119.
6. Socher, R.; Bauer, J.; Manning, C.; Ng, A. Parsing with compositional vector grammars. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria, 4–9 August 2013; pp. 455–465.
7. Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C.D.; Ng, A.; Potts, C. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 1631–1642.
8. Creutz, M.; Lagus, K. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.* **2007**, *4*, 1–34. [CrossRef]
9. Schuster, M.; Nakajima, K. Japanese and Korean Voice Search. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, Kyoto, Japan, 25–30 March 2012; pp. 5149–5152.
10. Chitnis, R.; DeNero, J. Variablelength word encodings for neural translation models. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 2088–2093.
11. Kunchukuttan, A.; Bhattacharyya, P. Orthographic syllable as basic unit for SMT between related languages. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 1912–1917.
12. Banerjee, T.; Bhattacharyya, P. Meaningless yet meaningful: Morphology grounded subword-level NMT. In Proceedings of the Second Workshop on Subword/Character Level Models, New Orleans, LA, USA, 5 June 2018; pp. 55–60.
13. Kudo, T. Subword regularization: Improving neural network translation models with multiple subword candidates. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; pp. 66–75.
14. Gage, P. A new algorithm for data compression. *C Users J.* **1994**, *12*, 23–38.
15. Larsson, N.J.; Moffat, A. Off-line dictionary-based compression. *Proc. IEEE* **2000**, *88*, 1722–1732. [CrossRef]
16. Karpinski, M.; Rytter, W.; Shinohara, A. An Efficient Pattern-Matching Algorithm for Strings with Short Descriptions. *Nord. J. Comput.* **1997**, *4*, 172–186.
17. Kida, T.; Matsumoto, T.; Shibata, Y.; Takeda, M.; Shinohara, A.; Arikawa, S. Collage system: A unifying framework for compressed pattern matching. *Theor. Comput. Sci.* **2003**, *298*, 253–272. [CrossRef]
18. Cormod, G.; Muthukrishnan, S. The string edit distance matching problem with moves. *ACM Trans. Algorithms* **2007**, *3*, 1–19. [CrossRef]
19. Jeż, A. A really simple approximation of smallest grammar. *Theor. Comput. Sci.* **2016**, *616*, 141–150. [CrossRef]
20. Takabatake, Y.; I, T.; Sakamoto, H. A Space-Optimal Grammar Compression. In Proceedings of the 25th Annual European Symposium on Algorithms, Vienna, Austria, 6–9 September 2017; pp. 1–15.
21. Gańczorz, M.; Gawrychowski, P.; Jeż, A.; Kociumaka, T. Edit Distance with Block Operations. In Proceedings of the 26th Annual European Symposium on Algorithms, Helsinki, Finland, 20–22 August 2018; pp. 33:1–33:14.
22. Lehman, E.; Shelat, A. Approximation algorithms for grammar-based compression. In Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, USA, 6–8 January 2002; pp. 205–212.

23. Lehman, E. Approximation Algorithms for Grammar-Based Data Compression. Ph.D. Thesis, MIT, Cambridge, MA, USA, 1 February 2002.
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the 31st Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
25. Sutskever, I.; Vinyals, O.; Le, V.Q. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 2 December 2014; pp. 3104–3112.
26. Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; Rush, A. OpenNMT: Open-source toolkit for neural machine translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 67–72.
27. Post, M. A call for clarity in reporting BLEU scores. In Proceedings of the Third Conference on Machine Translation, Brussels, Belgium, 31 October–1 November 2018; pp. 186–191.
28. He, X.; Haffari, G.; Norouzi, N. Dynamic Programming Encoding for Subword Segmentation in Neural Machine Translation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 3042–3051.
29. Deguchi, H.; Utiyama, M.; Tamura, A.; Ninomiya, T.; Sumita, E. Bilingual Subword Segmentation for Neural Machine Translation. In Proceedings of the 28th International Conference on Computational Linguistics, Online, 8–13 December 2020; pp. 4287–4297.
30. Fadaee, M.; Bisazza, A.; Monz, C. Data augmentation for low-resource neural machine translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 567–573.
31. Wang, X.; Pham, H.; Dai, Z.; Neubig, G. SwitchOut: An efficient data augmentation algorithm for neural machine translation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 856–861.
32. Ranto Sawai, R.; Paik, I.; Kuwana, A. Sentence Augmentation for Language Translation Using GPT-2. *Electronics* **2021**, *10*, 3082. [CrossRef]
33. Park, C.; Yang, Y.; Park, K.; Heuiseok Lim, H. Decoding Strategies for Improving Low-Resource Machine Translation. *Electronics* **2020**, *9*, 1562. [CrossRef]