

Beyond Junk Tokens: Subsumed Tokens in Japanese Subword Tokenization

Toshiki Yamaguchi and Kazuo Hara and Ikumi Suzuki

Graduate School of Science and Engineering

Yamagata University, Yamagata, Japan

{totayu.647,kazuo.hara,suzuki.ikumi}@gmail.com

Abstract

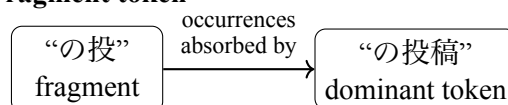
Subword tokenization is widely used to address the out-of-vocabulary problem in natural language processing. However, frequency-based subword tokenization can produce artifacts in the resulting vocabulary. Beyond the well-known junk tokens (meaningless fragments), we introduce the notion of *subsumed tokens*, valid words whose corpus frequencies are suppressed by longer tokens that contain them. Together with fragment tokens, these are referred to as *overlapping tokens*. We propose a post-processing method that reduces fragment tokens and restores subsumed tokens. The method searches over possible binary segmentations of character sequences and selects tokenizations that maximize corpus frequency, while optionally assigning bonuses to word spans identified by a morphological analyzer. As a post-processing step, the method can be applied to existing tokenizers such as Byte Pair Encoding (BPE) and the Unigram Language Model (ULM) without rebuilding their vocabularies. Although evaluated on Japanese, the approach may also be applicable to other languages given an appropriate morphological analyzer. Experiments on the Wiki-40B Japanese corpus show that our method reduces overlapping tokens, leading to more reliable statistics for valid words, and improves representation quality while preserving downstream performance across tasks.

1 Introduction

Tokenization converts raw text into sequences of tokens that form the basic units for representation learning in natural language processing (NLP). Traditional word-level tokenization suffers from the *out-of-vocabulary* (OOV) problem, which motivates subword methods such as Byte Pair Encoding (BPE) (Sennrich et al., 2016) and the Unigram Language Model (ULM) (Kudo, 2018). Because tokenization determines the units over which distributional statistics are collected, it can introduce

Overlapping tokens

Fragment token



Subsumed token

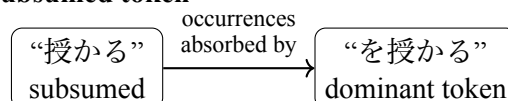


Figure 1: Two types of overlapping tokens arising in Japanese subword tokenization. Fragment tokens (top) are meaningless substrings that mainly occur as parts of longer tokens. Subsumed tokens (bottom) are valid words whose occurrences are suppressed because they are often absorbed into more frequent dominant tokens. Dominant tokens typically occur more frequently and absorb occurrences.

biases that affect learned representations across different tokenizers (e.g., BPE and ULM), which are typically driven by frequency- or likelihood-based objectives. Although subword tokenization alleviates the OOV problem, it can also introduce artifacts; for example, BPE is known to produce *junk tokens*—intermediate substrings with little semantic meaning that arise during the merge process (Bostrom and Durrett, 2020).

This paper focuses on Japanese as a representative case where token overlap is particularly pronounced, though similar phenomena arise in other languages without explicit word boundaries, such as Chinese.¹ In such languages, subword tokenization often produces overlapping tokens, where shorter tokens form substrings of longer ones. We refer to this as the **overlapping-token phenomenon**, where tokens overlap in the sense that shorter tokens form substrings of longer tokens, often reducing their independent occurrences in the tokenized corpus.

¹We observe similar subsumed-token patterns in Chinese, e.g., “高新技术” → “高新技术产业”.

Figure 1 illustrates how shorter tokens tend to be absorbed into longer dominant tokens under frequency- or likelihood-based tokenization. We distinguish two types of overlapping tokens. The first type is **fragment tokens**, meaningless substrings such as “の投” that appear only as parts of longer tokens (e.g., “の投稿” (*submission of*), “の投資” (*investment of*)). Fragment tokens correspond to the junk tokens reported in prior work. Such tokens rarely occur independently and therefore receive unreliable or poorly estimated representations. The second type is **subsumed tokens**, linguistically meaningful valid words such as “授かる” (*be blessed*)² whose occurrences are reduced because they are often incorporated into longer tokens such as “を授かる” (*be blessed with*), which absorb occurrences that would otherwise be assigned to the shorter token. Subsumed tokens extend fragment (junk) tokens by corresponding to valid words. As a result, such words may receive fewer observations during representation learning, potentially degrading their representations. This suggests that tokenization can distort the distributional statistics, affecting both intrinsic representation quality and downstream task performance. These overlaps can be viewed as structural artifacts of subword tokenization.

To address this issue, we propose a tokenization framework that reduces fragment tokens and restores subsumed tokens. The framework constructs binary trees over character sequences and selects segmentations that maximize corpus frequency while assigning bonuses to spans identified by a morphological analyzer. Unlike approaches that modify tokenizer training, our method operates as a post-processing step applicable to existing tokenizers such as BPE and ULM without modifying their vocabularies. Although our experiments focus on Japanese, the framework may also be applicable to other languages, particularly those without explicit word boundaries, given an appropriate morphological analyzer. Experiments on the Wiki-40B Japanese corpus show that the proposed method reduces overlapping tokens and improves embedding quality without degrading downstream performance.

Contributions.

- We formalize the notion of **overlapping to-**

²We refer to linguistically meaningful words as *valid words*. A formal operational definition used in our analysis is provided in Appendix A.

kens and distinguish two types: fragment tokens and subsumed tokens.

- We reveal the **subsumed token phenomenon**, in which valid words become systematically underrepresented due to frequency bias in subword tokenization.
- We propose a tokenizer-agnostic post-processing framework that reduces fragment tokens and restores subsumed tokens without modifying the tokenizer vocabulary, improving representation quality while preserving downstream performance.

2 Related Work

Subword tokenization and token artifacts.

Subword tokenization methods such as WordPiece (Wu et al., 2016), Byte Pair Encoding (BPE) (Sennrich et al., 2016), and the Unigram Language Model (ULM) (Kudo, 2018) are widely used to address the out-of-vocabulary problem. However, they can introduce artifacts in the resulting vocabulary. Bostrom and Durrett (2020) showed that BPE often produces so-called *junk tokens*, intermediate substrings with little semantic meaning.

Fragment tokens are closely related to such junk or scaffold tokens identified in prior work on BPE tokenization. In this paper, we distinguish two types of overlapping tokens: *fragment tokens*, meaningless substrings produced during tokenization, and *subsumed tokens*, valid words whose occurrences are suppressed by longer tokens. While fragment-like tokens have been discussed in prior work, subsumed tokens have received little attention.

Several studies attempt to reduce fragment-like artifacts by modifying the tokenizer training procedure. For example, Picky BPE (Chizhov et al., 2024) and Scaffold-BPE (Lian et al., 2025) discourage meaningless fragments during BPE construction. In contrast, our work does not modify tokenizer training but performs tokenization correction as a post-processing step. Moreover, we show that fragment tokens arise not only in BPE but also frequently in ULM tokenization.

BPE-Knockout (Bauwens and Delobelle, 2024) and Trimmed BPE (Cognetta et al., 2024) modify a BPE vocabulary after construction by removing undesirable subwords. In contrast, our method retains the original vocabulary and modifies only the segmentation, without permanently removing any

tokens. This also allows our method to be applied beyond BPE, including ULM.

Longer tokens across word boundaries. Recent studies have explored the use of longer tokens that span multiple words. Methods such as SuperBPE (Liu et al., 2025) and Boundless-BPE (Schmidt et al., 2025) report improvements in downstream performance by allowing multi-word tokens. Our work is complementary to this line of research. Rather than discouraging longer tokens, we highlight that when longer tokens dominate segmentation, shorter tokens corresponding to valid words may become underrepresented.

Japanese tokenization. Tokenization in Japanese is challenging because words are not separated by whitespace. A common strategy is to apply a morphological analyzer before subword tokenization. For example, Japanese BERT³ uses MeCab (Kudo et al., 2004) followed by WordPiece, while Japanese RoBERTa⁴ uses Juman++ (Morita et al., 2015) followed by ULM. Although morphological pre-segmentation can reduce certain tokenization artifacts by enforcing word boundaries, standard subword tokenization applied to raw Japanese text can still generate fragment tokens and subsumed tokens. This motivates the post-processing framework proposed in this work.

3 Motivation

Fragment tokens. Fragment tokens are substrings that appear as tokens but do not correspond to meaningful words.

Consider the Japanese phrase “論文の投稿” (*submission of a paper*). A linguistically valid segmentation would be

論文 | の | 投稿

However, a subword tokenizer may produce

論文 | の投 | 稿

Here the token “の投” is a fragment token. It does not form a valid word and carries little meaning in isolation. Such tokens arise because subword tokenizers learn tokens based on corpus statistics rather than linguistic boundaries.

³<https://github.com/cl-tohoku/bert-japanese>

⁴<https://huggingface.co/nlp-waseda/roberta-base-japanese>

Importantly, fragment tokens also arise in ULM; for example, “アニス” (244 occurrences) is less frequent than the longer token “ピアニスト” (*pianist*; 1752 occurrences). Other examples include “代記” (“年代記”, *chronicle*) and “コフスキー” (“チャイコフスキー”, *Tchaikovsky*).

Subsumed tokens. Subsumed tokens are valid words whose occurrences are often absorbed by longer tokens that cross word boundaries.

Consider the phrase “子を授かる” (*be blessed with a child*). A tokenizer may learn the token “を授かる”, which crosses the boundary between the particle “を” and the verb “授かる”. As a result, the valid word “授かる” appears less frequently as an independent token and becomes effectively suppressed.

Unlike fragment tokens, subsumed tokens correspond to valid words, yet their representation quality may degrade because their occurrences are absorbed by longer tokens.

Fragment vs. subsumed tokens. These two phenomena illustrate different effects caused by overlapping tokens in the vocabulary. Fragment tokens introduce meaningless substrings, whereas subsumed tokens suppress valid words by favoring longer tokens.

Such effects arise naturally in frequency-based subword tokenization, where tokens that partially overlap may compete with each other in representing the same character spans. In essence, overlapping tokens compete with each other to represent the same character spans in the text. We refer to the systematic suppression of valid words by longer overlapping tokens as the *subsumed token phenomenon*. To analyze these effects more formally, we introduce the notion of overlapping tokens in the next section.

4 Problem Definition

Overlapping tokens. We introduce the notion of *overlapping tokens* for a subword vocabulary V . For a token $t \in V$, let $|t|$ denote the number of characters in t . A token t is called an *overlapping token* if $|t| > 1$ and there exists another token $t' \neq t$ in V such that t is a contiguous substring of t' and $\text{freq}_V(t) \leq \text{freq}_V(t')$, where $\text{freq}_V(t)$ denotes the empirical corpus frequency of token t in the corpus tokenized with vocabulary V .

Fragment and subsumed tokens. Among overlapping tokens, we distinguish two cases.

Algorithm 1 Construct Optimal Binary Tree over Character Spans

```
1: Input: Sentence  $s$  of length  $n$ , Vocabulary  $V$ 
2: Output: Optimal binary tree  $T$ 
3: Initialize  $DP[i][i+1] \leftarrow \text{score}(s[i : i+1])$  for all  $i$ 
4: for  $\ell \leftarrow 2$  to  $n$  do
5:   for  $i \leftarrow 0$  to  $n - \ell$  do
6:      $j \leftarrow i + \ell$ 
7:      $k^* \leftarrow \arg \max_{i < k < j} (DP[i][k] + DP[k][j])$ 
8:      $DP[i][j] \leftarrow \text{score}(s[i : j]) + DP[i][k^*] + DP[k^*][j]$ 
9:      $\text{BackPtr}[i][j] \leftarrow k^*$ 
10:   end for
11: end for
12: Recover  $T$  from  $\text{BackPtr}$ 
13: return  $T$ 
```

Algorithm 2 Extract Token Segmentation from Binary Tree

```
1: Input: Binary tree  $T$ , Vocabulary  $V$ 
2: Output: Token sequence  $\text{token\_seq}$ 
3: Initialize  $\text{token\_seq} \leftarrow []$ 
4: function  $\text{Traverse}(\text{node})$ 
5:   if  $\text{span}(\text{node}) \in V$  then
6:     Append  $\text{span}(\text{node})$  to  $\text{token\_seq}$ 
7:   else
8:      $\text{Traverse}(\text{node.left})$ 
9:      $\text{Traverse}(\text{node.right})$ 
10:  end if
11: end function
12:  $\text{Traverse}(\text{root}(T))$ 
13: return  $\text{token\_seq}$ 
```

A token is called a *fragment token* if it does not correspond to a valid word.

A token is called a *subsumed token* if it corresponds to a valid word but its occurrences are largely absorbed by longer tokens that cross word boundaries.

Our goal is to reduce fragment tokens and subsumed tokens through a post-processing step applied to an existing tokenizer.

The detailed categorization procedure used in the experiments is described in Appendix A.

5 Method

We propose a post-processing method that refines token segmentation while retaining the vocabulary of an existing tokenizer. Unlike methods that modify vocabulary construction, our approach operates

at the segmentation stage without rebuilding the vocabulary.

The key idea is to prefer tokenizations composed of frequent tokens in the vocabulary and tokens corresponding to word spans identified by a morphological analyzer. This objective discourages fragment tokens and restores subsumed tokens defined in Section 4.

Given an input sentence, tokenization candidates are represented as binary segmentation trees over character spans. Each span receives a score based on its corpus frequency, and word spans identified by the morphological analyzer receive an additional bonus.

We formulate tokenization as a search over binary segmentation trees and compute the optimal structure using a dynamic programming algorithm similar to the inside algorithm used in parsing, allowing globally optimal tokenization to be found efficiently. Each node represents a character span, and the best split point is chosen to maximize the total score, enabling efficient search over exponentially many tokenization candidates.

Let W denote the set of word spans obtained by applying a morphological analyzer such as MeCab to the input sentence. Let $\text{freq}_V(t)$ denote the number of occurrences of token t in the training corpus tokenized using vocabulary V .

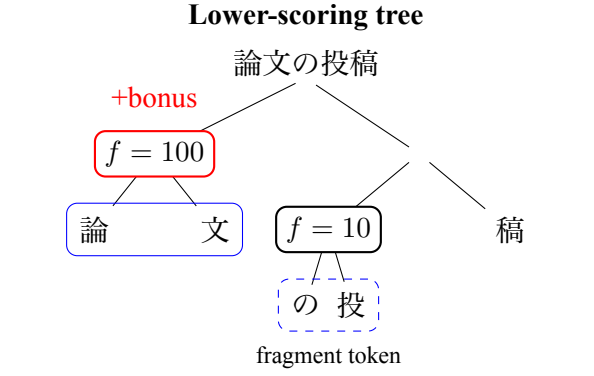
We assign the following score to each span t :

$$\text{score}(t) = \begin{cases} \text{freq}_V(t) + \text{bonus} & t \in V \cap W \\ \text{freq}_V(t) & t \in V \\ 0 & \text{otherwise} \end{cases}$$

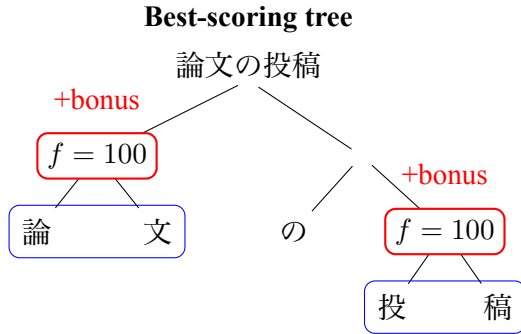
Each span contributes its score once in the tree, so the total score of a tree corresponds to the sum of the scores of its spans. The bonus is applied only when a span corresponds to both a vocabulary token and a word span identified by the morphological analyzer.

This dynamic programming procedure runs in $O(n^3)$ time with respect to sentence length. Algorithm 1 describes the dynamic programming procedure, and Algorithm 2 extracts the final token sequence.

Effect on token distribution. Although our method keeps the vocabulary fixed, modifying segmentation intentionally changes the token-frequency distribution. This redistribution reduces fragment-token usage and restores the frequency of valid subsumed tokens; it is therefore a core mechanism of our method.



(a) Lower-scoring tree containing the fragment token “の投”. Score = $\text{freq}(\text{“論文”}) + \text{freq}(\text{“の投”}) + \text{bonus} = 110 + \text{bonus}$.



(b) Best tree restoring the valid token “投稿”. Score = $\text{freq}(\text{“論文”}) + \text{freq}(\text{“投稿”}) + 2 \times \text{bonus} = 200 + 2 \times \text{bonus}$.

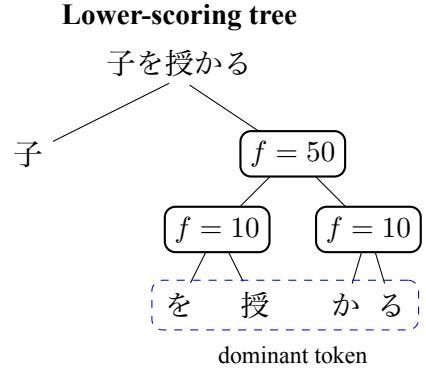
Figure 2: Comparison of a lower-scoring tree and the best tree for the phrase “論文の投稿”. The lower-scoring tree contains the fragment token “の投”, while the highest-scoring tree selects the valid tokens “論文” and “投稿”. Blue boxes indicate token spans. Dashed blue boxes indicate problematic tokens (e.g., fragment tokens). Red boxes mark tokens that receive a bonus (i.e., word spans identified by MeCab in the sentence).

6 Examples

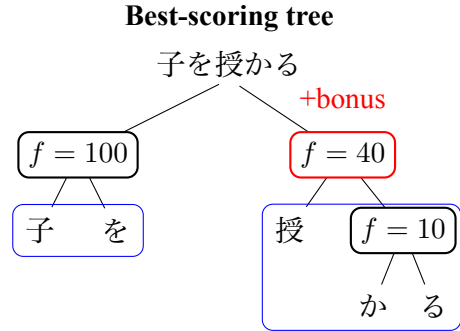
Eliminating a fragment token. Figure 2 compares two candidate trees for the phrase “論文の投稿”. In the lower-scoring tree, the fragment token “の投” is selected, resulting in a lower score. The proposed method instead prefers the tree that produces the valid tokens “論文” and “投稿”.

Restoring a subsumed token. Figure 3 shows the phrase “子を受かる”. In the lower-scoring tree, the longer token “を受かる” is selected. Our method instead restores the valid token “授かる”, which is identified as a word span by MeCab and receives a bonus.

These examples illustrate how the proposed scoring scheme discourages fragment tokens and restores valid tokens that would otherwise be subsumed by longer tokens.



(a) Lower-scoring tree where the token “授かる” is subsumed by the larger span “を受かる”. Score = $\text{freq}(\text{“を受かる”}) + \text{freq}(\text{“を授”}) + \text{freq}(\text{“かる”}) = 70$.



(b) Best tree restoring the token “授かる”. Score = $\text{freq}(\text{“子を”}) + \text{freq}(\text{“授かる”}) + \text{freq}(\text{“かる”}) + \text{bonus} = 150 + \text{bonus}$.

Figure 3: Comparison of a lower-scoring tree and the best tree for the phrase “子を受かる”. The lower-scoring tree selects the dominant token “を受かる”, which subsumes the valid token “授かる”, while the best tree restores the valid token “授かる”. Boxes and colors follow the same conventions as in Figure 2.

7 Experiment

Experimental settings. Our framework consists of a frequency-based tree construction, optionally augmented with a bonus for word spans identified by MeCab. To isolate the effect of the bonus term, we evaluate both variants.

We consider two tokenization settings depending on whether morphological segmentation is applied before subword tokenization (Figure 4). These settings allow us to analyze tokenization behavior both without external word boundaries and within a standard pipeline, and to evaluate whether tokenization artifacts can be mitigated by post-processing alone, without modifying the tokenizer or its vocabulary.

Setting 1 (Direct tokenization). Sentences are tokenized directly using a base subword tokenizer, without external morphological segmentation. This setting allows us to analyze how each

method handles segmentation purely through subword modeling. We compare:

- **Base:** BPE or ULM,
- **Ours(Base):** our post-processing applied to the base tokenizer,
- **Ours(Base)+B:** our method with an additional bonus for word spans identified by MeCab.

Setting 2 (Pre-tokenization). Sentences are first segmented using MeCab⁵ and then tokenized using a subword tokenizer. This reflects a typical Japanese NLP pipeline with explicit morphological boundaries. We compare:

- **MeCab→Base,**
- **Ours(MeCab→Base).**

Because MeCab segmentation already provides valid word boundaries, the bonus term is not applied in this setting. Consequently, subsumed tokens rarely occur, although fragment tokens may still appear.

Together, these settings allow us to evaluate how the proposed method reduces fragment tokens and restores subsumed tokens under different preprocessing conditions.

Corpus and evaluation. We use the Japanese portion of the Wiki-40B corpus (Guo et al., 2020), a cleaned and filtered Wikipedia dataset. The corpus is split into two halves: one half is used for training the tokenizer vocabulary, and the other half is used for evaluation, including token statistics, embedding training, and BERT pretraining for downstream evaluation.

Implementation details. BPE and ULM tokenizers are trained using SentencePiece (Kudo and Richardson, 2018).⁶ The vocabulary size $|V|$ ranges from 50K to 200K, following prior work suggesting benefits of larger vocabularies (Takase et al., 2025).

For Setting 2 pre-tokenization and for identifying word spans in each sentence for bonus scoring, we use MeCab with the *ipadic* dictionary.

For the bonus used in the proposed method, we set

$$\text{bonus} = \max_{t \in V} \text{freq}_V(t),$$

⁵<https://taku910.github.io/mecab>

⁶<https://github.com/google/sentencepiece>

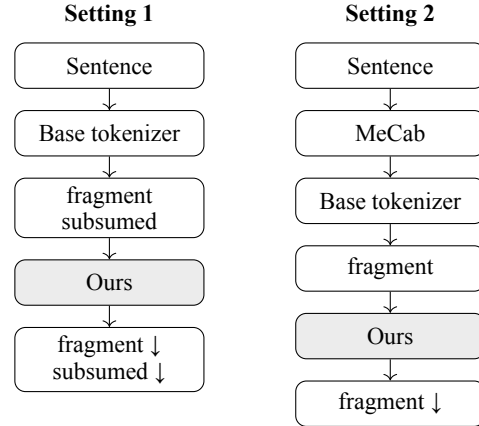


Figure 4: Overview of the tokenization pipelines used in our experiments. In Setting 1, direct subword tokenization may produce both fragment tokens and subsumed tokens (tokens that fully contain other tokens). Our method reduces such overlapping tokens. In Setting 2, MeCab segmentation prevents subsumed tokens, so the method mainly reduces fragment tokens.

where $\text{freq}_V(t)$ denotes the corpus frequency of token t under the base tokenizer. This value corresponds to the largest token frequency in the corpus. Intuitively, this provides a natural scale for the scoring function without introducing an additional hyperparameter. As a result, word spans identified by the morphological analyzer receive a strong preference during tokenization correction. The bonus is computed separately for each base tokenizer from the corresponding tokenized training corpus.

8 Results

We evaluate tokenization quality using three complementary measurements:

- the number of overlapping tokens (fragment tokens and subsumed tokens),
- intrinsic embedding quality,
- downstream task performance.

Overlapping tokens. We first analyze the structural behavior of the tokenizers in terms of overlapping tokens, before evaluating embedding quality and downstream performance. Specifically, we measure how frequently fragment tokens and subsumed tokens appear when tokenizing the evaluation corpus. Fragment tokens correspond to meaningless substrings that arise from over-segmentation, whereas subsumed tokens are valid tokens whose occurrences are reduced because

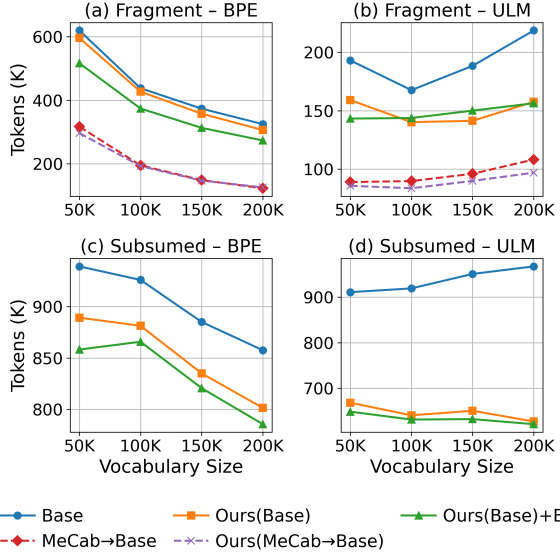


Figure 5: Fragment and subsumed token statistics across vocabulary sizes. The proposed framework consistently reduces both types of overlapping tokens for BPE and ULM.

they are absorbed into longer tokens. Lower values therefore indicate better tokenization quality. Figure 5 summarizes the results.

In Setting 1, both BPE and ULM produce a large number of fragment tokens and subsumed tokens. BPE generates many instances of both types, indicating that frequency-based tokenization often produces both meaningless fragments and nested token structures. ULM exhibits a different pattern: it produces substantially fewer fragment tokens than BPE, suggesting that its probabilistic segmentation tends to avoid excessive fragmentation. However, the number of subsumed tokens remains comparable to BPE, indicating that nested token structures still frequently occur.

The proposed method successfully reduces both fragment tokens and subsumed tokens for both tokenizers. In most cases the reductions are larger when the bonus scoring mechanism is enabled, except for the fragment-token counts of ULM, which are already relatively small.

In Setting 2, where MeCab provides explicit morphological boundaries, the number of overlapping tokens decreases substantially for all tokenizers. Both fragment tokens and subsumed tokens are roughly halved compared with Setting 1, although a non-negligible number of fragment tokens still remain. Under this setting, the proposed method yields modest but consistent improvements, slightly reducing fragment tokens for BPE and producing small additional reductions for

ULM. Subsumed tokens rarely appear in Setting 2 because explicit word boundaries prevent nested token structures, and therefore these results are not reported.

Overall, the results indicate that the proposed framework consistently mitigates the two major sources of token overlap—fragmentation and subsumption—across different tokenizers and pre-processing settings.

Intrinsic tasks. We evaluate intrinsic embedding quality using the JWSAN dataset (Inohara and Utsumi, 2022), which contains Japanese word pairs annotated with human similarity and relatedness ratings.

For each tokenization method, we train token embeddings using word2vec (Mikolov et al., 2013). We evaluate 395 word pairs that appear across all vocabularies ($|V| = 50K-200K$). Performance is measured by the correlation between human ratings and cosine similarity of token embeddings. Since word2vec training involves randomness, we report the average correlation over 10 independent runs, using the same random seed across tokenization methods within each run. Statistical significance is assessed using a one-sided paired t -test on the correlation scores from the corresponding runs, comparing each proposed method with its corresponding baseline ($p < 0.05$).

Table 1 reports the results. Overall, correlation scores in Setting 2 are slightly lower than those in Setting 1 for both similarity and relatedness. This suggests that embeddings trained with explicit MeCab pre-tokenization capture semantic relations somewhat less effectively in this dataset.

In Setting 1, the proposed method—especially the variant with bonus scoring—often improves the correlation scores across vocabulary sizes and tokenizers. These improvements are consistent with the analysis in Section 4, where subsumed tokens are shown to reduce the effective frequency of valid words during representation learning.

In contrast, improvements are generally small in Setting 2, where subsumed tokens rarely occur after MeCab pre-tokenization. This result indicates that the benefit of the proposed method mainly arises from restoring subsumed tokens that would otherwise be absorbed by longer tokens in raw sub-word tokenization.

Downstream tasks. We evaluate downstream performance using the JapaneseTopicWSD

Vocab	Method	Similarity				Relatedness			
		Setting 1		Setting 2		Setting 1		Setting 2	
		BPE	ULM	BPE	ULM	BPE	ULM	BPE	ULM
50K	Base	0.640	<u>0.663</u>	0.610	0.606	0.688	<u>0.716</u>	<u>0.658</u>	0.657
	Ours	0.643	0.661	<u>0.613</u>	<u>0.611*</u>	0.688	0.703	0.655	<u>0.659</u>
	Ours+B	<u>0.650*</u>	0.662	–	–	<u>0.694*</u>	0.707	–	–
100K	Base	0.651	0.659	0.617	0.615	0.697	0.713	<u>0.669</u>	0.664
	Ours	0.649	0.664*	<u>0.618</u>	0.615	0.696	0.708	0.667	0.664
	Ours+B	<u>0.664*</u>	<u>0.669*</u>	–	–	<u>0.708*</u>	<u>0.716</u>	–	–
150K	Base	0.659	0.648	<u>0.620</u>	0.622	0.711	0.704	0.669	<u>0.673</u>
	Ours	0.661	0.670*	0.619	0.622	0.711	0.716*	<u>0.671</u>	0.671
	Ours+B	<u>0.667*</u>	<u>0.672*</u>	–	–	<u>0.716*</u>	<u>0.723*</u>	–	–
200K	Base	0.656	0.631	0.622	0.621	0.709	0.693	0.675	0.673
	Ours	<u>0.657</u>	<u>0.661*</u>	<u>0.624</u>	<u>0.625*</u>	<u>0.710</u>	0.704*	<u>0.677</u>	<u>0.675</u>
	Ours+B	<u>0.657</u>	<u>0.661*</u>	–	–	<u>0.710</u>	<u>0.713*</u>	–	–

Table 1: Intrinsic evaluation on the JWSAN dataset. Similarity and relatedness scores represent the correlation between human judgments and cosine similarity of word2vec embeddings. Best results within each vocabulary size are shown in bold and underline. * indicates a significant improvement over the corresponding baseline ($p < 0.05$). Setting 2 does not include the bonus variant because McCab pre-tokenization already provides word boundaries, making the bonus term unnecessary.

Vocab	Method	Setting 1		Setting 2	
		BPE	ULM	BPE	ULM
50K	Base	<u>0.659</u>	0.658	0.672	<u>0.682</u>
	Ours	0.656	0.665	<u>0.675</u>	0.673
	Ours+B	0.654	<u>0.668</u>	–	–
100K	Base	0.648	0.653	0.665	0.658
	Ours	0.655*	0.660	<u>0.681*</u>	<u>0.662</u>
	Ours+B	<u>0.677*</u>	<u>0.671</u>	–	–
150K	Base	0.649	0.656	0.669	<u>0.679</u>
	Ours	0.650	0.648	<u>0.672</u>	0.663
	Ours+B	<u>0.652</u>	<u>0.659</u>	–	–
200K	Base	0.656	0.654	0.670	<u>0.682</u>
	Ours	0.655	<u>0.663</u>	<u>0.675*</u>	0.677
	Ours+B	<u>0.657</u>	0.647	–	–

Table 2: Macro F1 scores on the Japanese WSD task (token-level evaluation). Best results within each vocabulary size are shown in bold and underline. * indicates a significant improvement over the corresponding baseline ($p < 0.05$).

dataset.⁷ The dataset consists of sentences annotated with both word senses for target words and topic labels for sentences.

We conduct two tasks. For Word Sense Disambiguation (WSD), we select sentences containing target words whose senses appear more than 10 times in the dataset. This results in 13,040 sentences covering 149 word senses. For sentence classification, we select sentences whose topic labels appear more than 10 times, resulting in 25,665

Vocab	Method	Setting 1		Setting 2	
		BPE	ULM	BPE	ULM
50K	Base	0.633	0.639	0.642	0.635
	Ours	0.629	0.637	<u>0.647</u>	<u>0.641</u>
	Ours+B	<u>0.635</u>	<u>0.642</u>	–	–
100K	Base	0.620	0.625	0.646	<u>0.642</u>
	Ours	0.617	<u>0.640*</u>	<u>0.658</u>	0.641
	Ours+B	<u>0.629</u>	0.631	–	–
150K	Base	0.611	<u>0.637</u>	<u>0.656</u>	<u>0.637</u>
	Ours	0.618	0.630	0.640	0.636
	Ours+B	<u>0.635*</u>	0.633	–	–
200K	Base	0.604	0.621	0.642	0.643
	Ours	0.614*	<u>0.627</u>	<u>0.649</u>	<u>0.644</u>
	Ours+B	<u>0.622*</u>	0.617	–	–

Table 3: Macro F1 scores on the sentence classification task (sequence-level evaluation). Best results within each vocabulary size are shown in bold and underline. * indicates a significant improvement over the corresponding baseline ($p < 0.05$).

sentences covering 160 topic labels.

For both tasks, we perform 10 runs with stratified random train/dev/test splits, using the same split across tokenization methods within each run. For each method and run, we pretrain BERT (Devlin et al., 2019) on Wiki-40B Japanese tokenized by that method, fine-tune it on the downstream task, and report macro F1. We report averages over the 10 runs and assess significance using a one-sided paired t -test ($p < 0.05$). Detailed BERT training configurations are provided in Appendix B.

⁷<https://github.com/nut-jnlp/JapaneseTopicWSD>

Tables 2 and 3 present the results. In Setting 1, the proposed method—particularly the bonus variant—often yields higher macro F1 scores for both tasks. In Setting 2, improvements are generally smaller. This is especially noticeable for ULM-based tokenizers, where fragment tokens are already relatively rare.

Overall, the results suggest that reducing overlapping tokens (fragment tokens and subsumed tokens) can contribute to improved downstream performance when tokenization is applied directly to raw sentences.

Compression and Rényi entropy. To characterize the distributional effects of our method, we additionally examine compression and Rényi entropy. We measure compression by the total number of tokens after tokenization (fewer is better). In Setting 1, adding the word bonus increases the token count over the baseline by 0.79–2.20% across BPE and ULM with different vocabulary sizes, indicating a slight loss in compression efficiency.

Rényi entropy is defined as $H_\alpha(p) = \frac{1}{1-\alpha} \log \sum_i p_i^\alpha$, where p_i is the probability of token i , and smaller α gives relatively more weight to low-probability tokens, whereas larger α emphasizes high-probability tokens. With the word bonus, Rényi entropy consistently decreases at $\alpha = 0.5$ and increases at $\alpha = 3$ across all BPE and ULM vocabulary settings. This opposite behavior indicates that the redistribution of token frequencies affects low- and high-probability regions differently. Thus, reducing overlapping-token effects does not necessarily coincide with optimizing compression or Rényi entropy at a particular α .

9 Conclusion

We analyzed overlapping tokens—fragment tokens and subsumed tokens—that arise in frequency-based subword tokenization. We showed that these overlaps can be interpreted as structural artifacts of tokenization and proposed a post-processing method based on span scoring to mitigate them. Experiments demonstrated that the method consistently reduces overlapping tokens and can improve embedding quality and downstream task performance.

Limitations

Our method requires an $O(n^3)$ dynamic programming step over character spans, making tokeniza-

tion substantially slower than standard subword tokenization. On an AMD Ryzen 9 9950X CPU with 132GB RAM, tokenizing 10K sentences took on average across 10 runs 0.15s for BPE, 0.14s for ULM, and 13.5s for our method (12.8s for tree construction, 0.58s for tree traversal, and 0.12s for MeCab segmentation). The overhead is incurred during tokenization and does not affect model computation after tokenization. While the runtime was manageable in our sentence-level experiments, the cubic complexity may become costly for long texts without sentence boundaries. Sentence-level processing and parallelization could mitigate this overhead, but improving tokenization efficiency remains an important direction for future work.

Another limitation is that the current framework assumes tokenizers that produce overlapping tokens through frequency-based or likelihood-based subword segmentation, such as BPE and ULM. Other tokenization schemes with different design principles may exhibit different overlap patterns, and further study is needed to evaluate the applicability of our approach.

Finally, our experiments focus on Japanese datasets and a limited set of intrinsic and downstream tasks. Although our method may be applicable to other languages given an appropriate morphological analyzer, its effectiveness across languages, corpora, and large-scale pretraining settings remains to be evaluated.

References

- Thomas Bauwens and Pieter Delobelle. 2024. [BPE-knockout: Pruning pre-existing BPE tokenisers with backwards-compatible morphological semi-supervision](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5810–5832, Mexico City, Mexico. Association for Computational Linguistics.
- Kaj Bostrom and Greg Durrett. 2020. [Byte pair encoding is suboptimal for language model pretraining](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.
- Pavel Chizhov, Catherine Arnett, Elizaveta Korotkova, and Ivan P. Yamshchikov. 2024. [BPE gets picky: Efficient vocabulary refinement during tokenizer training](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16587–16604, Miami, Florida, USA. Association for Computational Linguistics.

- Marco Cognetta, Tatsuya Hiraoka, Rico Sennrich, Yuval Pinter, and Naoaki Okazaki. 2024. [An analysis of BPE vocabulary trimming in neural machine translation](#). In *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 48–50, Mexico City, Mexico. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. 2020. [Wiki-40B: Multilingual language model dataset](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2440–2452, Marseille, France. European Language Resources Association.
- Keisuke Inohara and Akira Utsumi. 2022. Jwsan: Japanese word similarity and association norm. *Language Resources and Evaluation*, 56(1):109–137.
- Taku Kudo. 2018. [Subword regularization: Improving neural network translation models with multiple subword candidates](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.
- Taku Kudo and John Richardson. 2018. [SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.
- Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. [Applying conditional random fields to Japanese morphological analysis](#). In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.
- Haoran Lian, Yizhe Xiong, Jianwei Niu, Shasha Mo, Zhenpeng Su, Zijia Lin, Hui Chen, Jungong Han, and Guiguang Ding. 2025. [Scaffold-bpe: Enhancing byte pair encoding for large language models with simple and effective scaffold token removal](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):24539–24548.
- Alisa Liu, Jonathan Hayase, Valentin Hofmann, Se-woong Oh, Noah A Smith, and Yejin Choi. 2025. [SuperBPE: Space travel for language models](#). In *Second Conference on Language Modeling*.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Efficient estimation of word representations in vector space](#). *Preprint*, arXiv:1301.3781.
- Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. [Morphological analysis for unsegmented languages using recurrent neural network language model](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2292–2297, Lisbon, Portugal. Association for Computational Linguistics.
- Craig W. Schmidt, Varshini Reddy, Chris Tanner, and Yuval Pinter. 2025. [Boundless byte pair encoding: Breaking the pre-tokenization barrier](#). In *Second Conference on Language Modeling*.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Neural machine translation of rare words with subword units](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Sho Takase, Ryokan Ri, Shun Kiyono, and Takuya Kato. 2025. [Large vocabulary size improves large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1015–1026, Vienna, Austria. Association for Computational Linguistics.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, and 11 others. 2016. [Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation](#). *arXiv.org*, pages 1–23.

A Categorization of Overlapping Tokens

To quantify the artifacts discussed in the main text, we categorize overlapping tokens into fragment, subsumed, and others using the procedure described below.

In the main text we introduce an intuitive distinction between *fragment tokens* and *subsumed tokens*. For quantitative analysis we additionally introduce a third category, *others*.

Dominant tokens. For an overlapping token t , we define its *dominant tokens* as the set

$$D(t) = \{t' \mid \begin{array}{l} t \text{ is a substring of } t', \\ \text{freq}_V(t) \leq \text{freq}_V(t') \end{array}\}$$

Here $\text{freq}_V(\cdot)$ denotes the token frequency in the tokenized corpus with vocabulary V used in our experiments.

That is, a dominant token is a token that contains t as a contiguous substring and is at least as frequent as t .

Morphological segmentation. Let $\text{list}(t)$ denote the token sequence obtained by segmenting t using MeCab. If the segmentation contains an unknown token (UNK), the token is treated as containing an unknown word. A token t is considered a *valid word* when

$$\text{UNK} \notin \text{list}(t).$$

Operationally, this means that t is considered valid if MeCab can segment it entirely into known words, possibly consisting of multiple words.

Subsumption. For a dominant token t' , we say that t is *subsumed* by t' when

$$\text{list}(t) \sqsubset \text{list}(t')$$

i.e., the morphological sequence of t appears as a contiguous subsequence of $\text{list}(t')$.

Motivation. The categorization rules aim to distinguish two types of overlapping tokens. A token is categorized as *subsumed* when it forms a valid word that appears as part of a longer token. In contrast, a token is categorized as *fragment* when it corresponds to an incomplete morphological unit. Tokens that do not clearly fall into either case are assigned to the *others* category.

Token categorization. For each dominant token $t' \in D(t)$ we first assign a temporary category using Algorithm 3. The final category of t is determined as follows:

- if all dominant tokens categorize t as *fragment*, then t is categorized as *fragment*;
- if all dominant tokens categorize t as *subsumed*, then t is categorized as *subsumed*;
- otherwise t is categorized as *others*.

These rules approximate the linguistic distinction between incomplete fragments and valid words overshadowed by longer tokens.

Algorithm 3 Pairwise categorization for (t, t')

```

1: Input: overlapping token  $t$ , dominant token  $t'$ 
2: Output: temporary category
3: if  $\text{list}(t) \sqsubset \text{list}(t')$  then
4:   return subsumed
5: end if
6: if  $|\text{list}(t)| = 1$  and  $\text{UNK} \notin \text{list}(t)$  then
7:   return others
8: end if
9: if  $\text{UNK} \in \text{list}(t')$  then
10:  return others
11: end if
12: return fragment

```

Examples.

- $t = \text{授かる}, t' = \text{を授かる}$
 $\text{list}(t) = [\text{授かる}]$
 $\text{list}(t') = [\text{を}, \text{授かる}]$
 $\rightarrow$ subsumed
- $t = \text{子}, t' = \text{子を}$
 $\text{list}(t) = [\text{子}]$
 $\text{list}(t') = [\text{子}, \text{を}]$
 $\rightarrow$ subsumed
- $t = \text{の投}, t' = \text{の投稿}$
 $\text{list}(t) = [\text{の}, \text{投}]$
 $\text{list}(t') = [\text{の}, \text{投稿}]$
 $\rightarrow$ fragment
- $t = \text{プター}, t' = \text{ヘリコプター}$
 $\text{list}(t) = [\text{UNK}]$
 $\text{list}(t') = [\text{ヘリコプター}]$
 $\rightarrow$ fragment
- $t = \text{プタ}, t' = \text{プター}$
 $\text{list}(t) = [\text{UNK}]$
 $\text{list}(t') = [\text{UNK}]$
 $\rightarrow$ others
- Example with multiple dominant tokens.
 $t = \text{本人}$
 $t'_1 = \text{本人の}, t'_2 = \text{日本人}$
 $\text{list}(t) = [\text{本人}]$
 $\text{list}(t'_1) = [\text{本人}, \text{の}]$
 $\text{list}(t'_2) = [\text{日本人}]$

The pair (t, t'_1) is categorized as subsumed, while (t, t'_2) is categorized as others. Since the temporary categories are inconsistent, the final category of t becomes others.

For reproducibility, all fragment and subsumed token counts reported in the experiments were computed using the categorization procedure described above.

B BERT Training Configuration

For each tokenization method, we pretrain a BERT model from scratch on the Wiki-40B Japanese corpus. The tokenizer corresponding to each method is applied to the corpus before training. All models follow the standard BERT base architecture with 12 transformer layers, 12 attention heads, and hidden size 768. Pretraining is performed using the masked language modeling (MLM) objective with masking probability 15%, maximum sequence length 128 tokens, and a mini-batch size of 80. Training is performed for 3 epochs with a learning rate of 1×10^{-4} . All tokenization methods use the same model architecture and training configuration to ensure fair comparison.

The pretrained BERT models are finetuned on the JapaneseTopicWSD dataset for the downstream tasks described in the main text. For Word Sense Disambiguation (WSD), we use a batch size of 4 and learning rate 2×10^{-5} , training for up to 15 epochs. For sentence classification, we use a batch size of 8 and learning rate 2×10^{-5} , training for up to 10 epochs.