Studies on Subword-based Low-Resource Neural Machine Translation: Segmentation, Encoding, and Decoding

Haiyue Song

February 2024

Abstract

In a world rich with diverse ideas and cultures, humans are isolated into islands of distinct languages. Machine translation (MT) serves as a bridge, facilitating information access and cross-lingual communication. Evolved from rule-based systems and statistical models, neural machine translation (NMT) is the state-of-the-art paradigm. While achieving near-human performance in high-resource languages, it is important to note that the majority of 7,100+ languages in the world are low-resource and have not benefitted significantly from the advancements in machine translation, leading to a substantial disparity among languages.

We improve the existing NMT systems to enhance the translation quality, especially for low-resource scenarios. In a conventional NMT framework, the embedding layer first converts tokens in the source language into embedding vectors, acting as a gate from the discrete natural language data to the continuous representation used in the neural model. The encoder then provides context-aware representations for the decoder, which then generates probability distribution for each next token and selects the token with the highest probability step by step, converting internal representations into a sentence in the target language. Though word or character is a more natural definition of the minimal input unit in the embedding layer, using subword is the default way because subwords better handle unseen words by segmenting them into seen subwords given a vocabulary of limited size, and bring higher efficiency than using characters by shorter input sequences. However, the artificial nature of subwords introduces new challenges in the segmentation, encoding, and decoding phases.

This thesis attempts to develop subword-based neural machine translation systems by answering three questions. **1)** One word can be segmented into different

subword segmentations. How do we determine the best one or best-k segmentation? **2)** The encoder converts subwords into contextual representations. How do we leverage multiple subword segmentations to capture the multi-perspective information of one word? **3)** The decoder generates a sequence of subwords in the target language as the translation. However, probability information from other subword segmentations that form the same word is underutilized. How do we incorporate subword segmentations that form the same word during decoding?

In Chapter 1, we provide an overview of the subword-based NMT model, including its structure, mechanism, and how subwords are used in the model. We emphasize the advantages together with the challenges brought by using subwords.

In Chapter 2, we propose SelfSeg, a neural subword segmenter that yields linguistically intuitive segmentation and is faster during training and decoding compared to previous neural methods. SelfSeg takes a word in the form of a partially masked character sequence as input, optimizes the word generation probability, and generates the segmentation with the maximum posterior probability, which is calculated using a dynamic programming algorithm. Additionally, we propose a regularization mechanism that allows the segmenter to generate best-k segmentations for one word. Moreover, it is trained in a self-supervised way that relies on only monolingual word-level data, making it applicable to low-resource languages without large-scale parallel resources.

In Chapter 3, we propose a BERT-based subword segmenter that generates subword segmentation that utilizes the contextualized semantic embeddings of words from the BERT model. During training, it maximizes the marginal probability from all possible segmentations of one word using a dynamic programming algorithm. During inference, it selects the one with the highest probability. Furthermore, we propose a probability-based regularization method that enables the segmenter to produce multiple segmentations for one word to improve the robustness of neural machine translation systems. Based on a pre-trained BERT encoder, it only requires little training data to achieve reasonable segmentations, making it especially applicable in low-resource scenarios.

In Chapter 4, we propose DiverSeg to exploit diverse segmentations from multiple subword segmenters that capture the various perspectives of each word in

the encoder. In DiverSeg, multiple segmentations are encoded using a subword lattice input, a subword-relation-aware attention mechanism integrates relations among subwords, and a cross-granularity embedding alignment objective enhances the similarity across different segmentations of a word. We found incorporating information from multiple aspects enhances the performance of neural machine translation, especially in low-resource scenarios.

In Chapter 5, we propose SubMerge, a decoding algorithm that merges the probabilities of multiple subword segmentations that form the same word, which we call equivalent segmentations. This is specially designed for the subword regularized NMT model. It leverages multiple subword segmentations of one target sentence during training as a data augmentation method, which is effective for low-resource scenarios. SubMerge is a nested search algorithm where the outer beam search treats the word as the minimal unit, and the inner beam search provides a list of word candidates and their probabilities, merging subword segmentations that form the equivalent word. It estimates the probability of the next word more precisely, providing better guidance during inference. We show it consistently outperforms the beam search algorithm in several low-resource MT datasets in terms of BLEU scores.

The methods proposed in this thesis have effectively addressed the issues in the subword-based NMT systems and significantly enhanced the translation performance especially in low-resource scenarios. In Chapter 6, we summarize the thesis and outline the possible directions for future work.

Acknowledgments

I would like to thank everyone who has supported and accompanied me during the Ph.D. journey at Kyoto University.

First of all, I extend my heartfelt gratitude to my supervisor, Professor Sadao Kurohashi, for his precious guidance and great patience throughout my whole Ph.D. course. I obtained insightful suggestions about the selection of research topics, paper writing, and making clear presentations. I always found his words philosophical, humanistic, and reflective of consideration of the impact on society, which refreshed my understanding of the motivation and the goal of doing research. He also provided various support in applying for scholarships and internships, and daily life support especially in the period of the pandemic.

I also would like to thank Dr. Raj Dabre and Associate Professor Chenhui Chu. Thank you for teaching me the basics of machine translation and the discussion of every idea in detail in the weekly meeting. Thanks for the support, encouragement, and patience during the struggling time in research and in life.

I am also grateful to Professor Tatsuya Kawahara and Professor Ko Nishino for serving as members of my doctoral committee, and thanks for the advice and comments to improve the quality of this thesis.

I wish to express my appreciation to all previous and current members in the laboratory. I would like to thank Dr. Fabien Cromieres, who led me into the machine translation field, Associate Professor Yugo Murawaki, Professor Daisuke Kawahara (currently at Waseda University), and Assistant Professor Fei Cheng, who provided insightful comments in the weekly discussion. Thank Dr. Tomohide Shibata (currently at Yahoo), Assistant Professor Yin-Jou Huang, and Assistant Professor Ribeka Tanaka (currently at Tokyo University of Technology) for their

advice and suggestions in the research meeting, which are the guiding light of my research journey. Especially, I would like to express my thanks to Dr. Hirokazu Kiyomaru, who is my mentor, senior, and friend, with whom I attended The Great Wall Marathon.

I am grateful to Zhuoyuan Mao, Zhen Wan, Weiqi Gu, and Dr. Qianying Liu (currently at Rinna), my friends and collaborators, with whom the journey became so much more colorful, and without whose collaboration many ideas could not be completed.

I wish to express my appreciation to Shuichiro Shimizu, Youyuan Lin, Dr. Arseny Tolmachev, Dr. Shuhei Kurita, Dr. Tareq Alkhaldi, and Frederic Bergeron for providing a comfortable research environment by installing and maintaining servers and external services. Special thanks to secretaries Terumi Kosugi, Yukimi Ishida, and Naho Yoshitoshi, who provided great help in the administrative procedures.

Thank you to Takashi Kodama, Kazumasa Omura, and Nobuhiro Ueda, with whom I share information on research and daily life in the same room. Thank Norio Takahashi, Takumi Yoshikoshi, Yu Tanaka, Ryota Nakao, Kohei Abe, Shengzhe Li, Leyi Yan, Dr. Yudai Kishimoto, Ryuichi Ozaki, Nozomu Karai, Boonnithi Jiaramaneepinit, Yihang Li, Zhengdong Yang, and Iglika Nikolova-Stoupak for the discussion about recent papers, research ideas, daily life events, and English learning.

I extend my sincere appreciation to my internship mentors and colleagues, Dr. Eiichiro Sumita, Dr. Masao Utiyama, Dr. Hideki Tanaka, Dr. Atsushi Fujita, Dr. Chenchen Ding, Dr. Chakrabarty Abhisek, Dr. Kaing Hour, Dr. Tran Van-Hien, Wadkar Sudarshan, for the tremendous support during my internship and work at NICT.

I am grateful to my family and friends for their endless support and understanding.

Lastly, I would like to thank Yahui Fu for being by my side during my PhD journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Translation serves to acquire, disseminate, and exchange information across different languages. For instance, in a tourist city like Kyoto, millions of foreign travelers acquire information written in Japanese and communicate with locals when navigating thousands of places of interest. While human translation is limited by its availability and cost, machine translation applications [88],[1] on the other hand, offer instant, easily accessible and low-cost solutions.

State-of-the-art neural machine translation achieves near-human performance in high-resource directions, such as English to Chinese. However, the performance is still unsatisfactory for translation directions involving low-resource languages. Moreover, the majority of 7,100+ languages in the world are low-resource languages that have not benefitted significantly from the advancements in machine translation. This leads to a substantial disparity of information acquisition and dissemination ability among languages, which further results in the disparity in regional development.

Subword-based neural machine translation systems use subwords as the minimal input and output unit. They show higher translation quality than word-based ones and higher efficiency than character-based ones [66], thus becoming the dominant paradigm. Despite the advantages, using subwords introduces challenges in

[1] `voicetra.nict.go.jp` and `mt-auto-minhon-mlt.ucri.jgn-x.jp`

the segmentation, encoding, and decoding phases in low-resource scenarios.

This thesis aims to tackle challenges within neural machine translation systems that use subwords, thereby enhancing translation quality, particularly in low-resource scenarios. The rest of this section begins with a brief history of machine translation from rule-based, statistical to neural systems, and machine translation in the large language model era. We then explain the architecture of the default subword-based neural machine translation system, the challenges brought by using subwords, and how we address them. Finally, we show the outline of this thesis.

1.2 History of Machine Translation

This section offers a brief history of machine translation and their ability in low-resource scenarios, including early rule-based systems, statistical machine translation systems and recent neural models. Additionally, we provide a perspective of how large language models further advance machine translation.

1.2.1 Early Rule-based Machine Translation Systems

Machine translation, defined as a system intended to perform translation by computer without human intervention, has been one of the central research topics in the natural language processing (NLP) field owing to its wide range of applications. The initial decade of machine translation was marked by high interest and support in the 1950s, driven by the goal of achieving high-speed and high-quality translation of texts. Notable early projects included Georgetown's GAT (Georgetown Automatic Translation) [95], which is one of the earliest machine translation projects developed by linguists and programmers. In the 1970s, SYSTRAN [133] emerged as one of the first marketed machine translation systems. It replaced earlier systems in various government agencies. Meanwhile, a machine translation system for science was developed in the 1960s in Japan that translates English titles of scientific and technical papers into Japanese [96, 97]. As a high-level application, the machine translation process involves knowledge of both the source language and the target language, including synonyms for words and phrases, grammar, and semantic knowledge. These early approaches usually

require significant effort using bilingual experts to craft the rules into programs.

However, because of the complexity of language, it is nearly impossible to represent all linguistical phenomenons into a limited number of rules. These systems are difficult to construct and only successful in several high-resource language pairs such as Russian→English and English→Japanese.

1.2.2 Statistical Machine Translation

Data-driven statistical approaches make it possible to learn knowledge from parallel bilingual text. In the early 1990s, Brown et al. [8] built a statistical machine translation (SMT) system that estimates the translation probability from one word in the source language into any particular word in the target language, with parameters in the model learned from parallel corpora. This system exceeded commercial-level translation quality between high-resource directions such as French and English at that time. With the development of statistical machine translation techniques, the translation quality keeps increasing and more low-resource language pairs are covered through the progress of multilingual parallel corpora construction [34].

However, a typical SMT system contains separate components, such as word aligners, translation rule extractors, and feature extractors, making it hard to locate the performance bottleneck and optimize the model automatically.

1.2.3 Neural Machine Translation

Neural machine translation (NMT) has shown state-of-the-art performance for a large number of language pairs and become the dominant paradigm for machine translation. It consists of tightly integrated Embedding Layer, Encoder and Decoder components, making end-to-end training possible through back-propagation optimization. Recurrent neural networks [36], long short-term memory [46] and gated recurrent [18] neural networks generate the hidden states of each input token computed from previous input tokens. Transformer architecture [135] leverages attention mechanisms [5] to enable high parallelization and achieves high-speed training and inference as well as high translation quality. It thus becomes the default architecture in machine translation and other NLP tasks.

Low-resource NMT is challenging because NMT models require substantial data and struggle with learning from sparse data. In this thesis, we define low-resource as those pairs with less than $100k$ parallel sentences and very-low-resource as those with less than $10k$ parallel sentences. In these scenarios, it requires careful tuning of hyper-parameters such as vocabulary size, dropout rate, batch size and so on to match the performance with statistical MT systems [114]. Other methods to improve low-resource NMT performance include incorporating linguistic feature [14, 13], exploiting multilingualism [22], pre-training leveraging monolingual data [78] or data from related languages as transfer learning [48, 120, 21], or simply creating high-quality corpora for the desired pairs or domains [109, 25].

1.2.4 Large Language Model for Machine Translation

Large-scale generative language models (LLMs) such as GPT-4 [99] and Llama 2 [134] have greatly propelled the research and application of natural language processing. It shows a high generalization ability to a large range of NLP tasks, including but not limited to machine translation, dialogue, question answering, and summarization [68, 141, 77]. However, these LLMs are biased towards English. When applied to the machine translation tasks, the translation quality for high-resource translation directions containing English is comparable to NMT models utilizing prompting techniques [151, 137]. However, the translation quality involving low-resource language is far from satisfactory compared to that of NMT models due to the imbalanced training data [94].

Prompting empowers LLM with great potential besides traditional fine-tuning or aligning [107] methods. Few-shot in-context learning [9] allows the model to quickly adapt to a new task or a new domain by showing several examples that are fixed or retrieved from a dataset by similarity. Chain-of-thought prompting [146] enables reasoning ability, which is crucial for complex tasks or samples. LLMs can also learn using tools to solve the domain-specific tasks [111] or collaborate with other LLM agnents [102]. With these, LLMs may achieve comparable performance with the traditional NMT paradigm and become the new paradigm in the future.

1.3 Subword-based Neural Machine Translation

1.3.1 Motivation of Using Subwords

Subwords refer to smaller units of language that are larger than individual characters but smaller than whole words, which is created by subword segmenter as shown in Figure 1.1. Different from *word* or *character*, *subword* is an artificial definition. This results in the ambiguity of subword segmentation, where one word can yield multiple different subword segmentations. For example, another subword segmentation of *watching* is *wat + ching*.

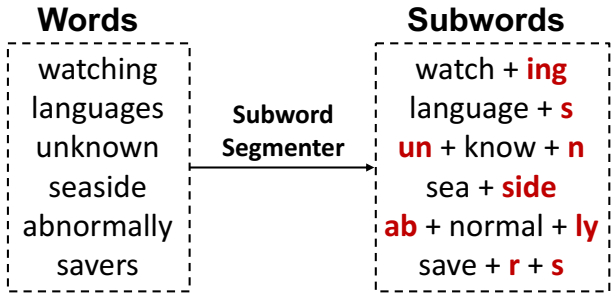


Figure 1.1: Words are segmented into subwords through subword segmenter.

Subword-based NMT is the dominant paradigm of NMT systems, where the minimal unit of input and output are subwords instead of words or characters. This is because using subwords brings advantages over using words or characters. Initial works on NMT used word-level vocabularies. As the vocabulary size significantly increases the parameter numbers in the embedding layers and computational cost in the softmax layer, the vocabulary size is usually constrained to tens of thousands. Therefore, the word vocabulary could only represent the most frequent words, leading to the out-of-vocabulary (OOV) problem [126]. Character-based approaches solve the OOV problem; however, they introduce higher computational complexity and translation latency because they generate longer sequences and require deeper-stacked models [39, 72, 16]. Using subwords in the vocabulary and embedding layers addresses both the OOV and the computational cost problems, thus becoming the default [116, 106, 61]. In this thesis, we pri-

marily use and improve the subword-based NMT architecture for low-resource NMT.

1.3.2 Overview of Subword-based Neural Machine Translation Architecture

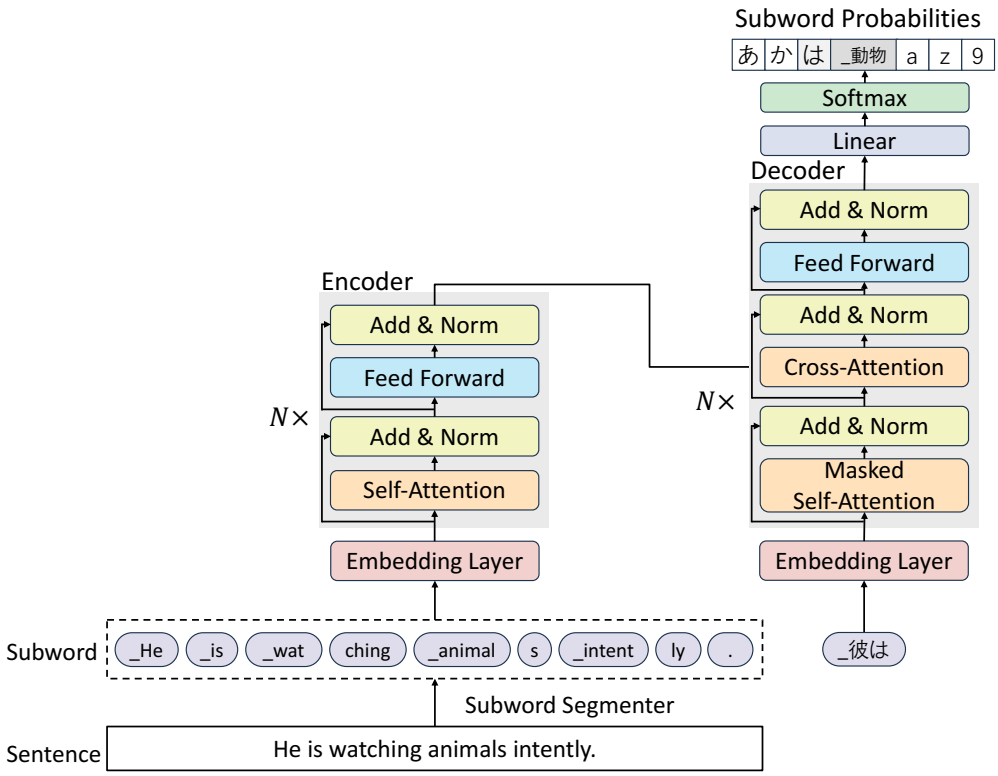


Figure 1.2: Overview of the Subword-based Neural Machine Translation Architecture. This shows the inference process. During training, the sentences in the target language is also segmented into subword sequences as supervision.

Figure 1.2 provides a general overview of the Subword-based NMT system. First, the subword segmenter converts input text into subwords as a pre-processing procedure. It is done for both the source sentence and target sentence during training and only done for the source sentence during inference. After that, each subword is converted into continuous embedding through an embedding layer.

The encoder then provides contextual embeddings for the decoder to predict the probabilities of each next subword in the target language, conditioned on all previously generated tokens. We can simply select the subword with the highest probability, as the greedy search does, or enlarge the searching space by considering multiple paths as in the beam search during inference. Finally, there is a simple post-editing procedure to combine the generated sequences of subwords into the output sentence.

1.3.3 Subword Segmentation and Embedding

A subword segmenter converts discrete textual data in natural language into a sequence of subwords. The embedding layer then converts each subword into a continuous embedding that the neural model can handle.

Let $\boldsymbol{x}_{1:T}$ denote a word that comprises T characters, that is $\boldsymbol{x}_{1:T} = (x_1, ..., x_T)$. Let $\boldsymbol{a}_{1:\tau_a}$ denote one segmentation of $\boldsymbol{x}_{1:T}$ that comprises τ_a subwords, that is $\boldsymbol{a}_{1:\tau_a} = (a_1, ..., a_{\tau_a})$. For each subword a_i in $\boldsymbol{a}_{1:\tau_a}$, it is non-empty substrings of $\boldsymbol{x}_{1:T}$ and in a predefined finite size subword vocabulary V, that is $a_i \in V$. The set of all valid segmentations for a word is represented as S_x, where $\forall \boldsymbol{a}, \boldsymbol{a} \in S_x$. A deterministic segmenter such as byte-pair encoding (BPE) [116] is an injective function that maps $\boldsymbol{x}_{1:T}$ to a specific $\boldsymbol{a}_{1:\tau_a}$ whereas a stochastic segmenter such as BPE-dropout [106] is a multivalued function that maps one word to a set of segmentations. The subword vocabulary contains characters and subwords. Therefore, the subword-based method handles rare words better by segmenting them into known subwords.

Subwords convert into embeddings in the embedding layers of the encoder and decoder. The embedding layer serves as a crucial component that bridges the gap between discrete textual data and continuous representations. Each subword a_i in the segmentation $\boldsymbol{a}_{1:\tau_a}$ is mapped to a high-dimensional vector $\boldsymbol{e}_i$. This mapping is accomplished through a lookup table where each unique subword in the vocabulary V has a corresponding vector. The vectors are typically initialized randomly and then learned and adjusted during the training process. The embedding layer in the encoder side also provides positional information through absolute positional encoding [135] or relative positional encoding [118].

1.3.4 Subword Encoding

The encoder maps individual subword representations to contextual representations. The input to the encoder is a sequence of subword representations $\boldsymbol{e}$ and the output is a sequence of contextual representations $\boldsymbol{z}$. For each layer l in the encoder, the transformation is defined as:

$$\boldsymbol{z}^l = \text{EncoderLayer}(\boldsymbol{z}^{l-1}) \tag{1.1}$$

where $\boldsymbol{z}^0 = \boldsymbol{x}$.

The attention mechanism is the key component in the encoder and decoder. In its simplest form, attention can be described as mapping a query and a set of key-value pairs to an output, calculated as a weighted sum of the values. The weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.2}$$

This formulation allows the model to focus on different parts of the input sequence, which is essential for tasks like translation, where the relevance of input elements can vary.

1.3.5 Subword Decoding

Sentence in the target language is generated from the decoder utilizing a decoding algorithm.

The decoder takes the final state of the encoder as the initial input, which encapsulates the semantical information from the sentence in the source language. In each step t, the decoder is auto-regressive, consuming the previously generated symbols as additional input. The transformation in each decoder layer l (L layers in total) can be described as:

$$\boldsymbol{y}_t^l = \text{DecoderLayer}(\boldsymbol{y}_t^{l-1}, \boldsymbol{z}) \tag{1.3}$$

where $\boldsymbol{y}_t^0$ is the embedding of the t-th symbol of the output sequence and $\boldsymbol{z}$ is the output of the encoder. Following the final decoder layer, the output $\boldsymbol{y}_t^L$ is

converted to $\boldsymbol{v}_t$ through a linear layer that maps the decoder's high-dimensional space to the size of the target subword vocabulary. Subsequently, a softmax layer converts it into a probability distribution over the subword vocabulary $\boldsymbol{p}_t$.

The decoding algorithm decides how to generate the translated output. The algorithm takes the probability distribution of each subword in the target vocabulary $\boldsymbol{p}_t$ as input. We briefly describe two primary strategies: greedy search and beam search. (1) **Greedy search** selects the subword with the highest probability as the next output token at each step. Let $\boldsymbol{y}_{1:t-1}$ be the sequence of subwords generated so far, and $P(y_t|\boldsymbol{y}_{1:t-1})$ be the probability of a subword a being the next subword. The next subword y_t in the sequence is chosen by:

$$y_t = \underset{a \in V}{\operatorname{argmax}} \; P(a|\boldsymbol{y}_{1:t-1}) \tag{1.4}$$

This process is repeated until a special end-of-sequence token is generated or a maximum length is reached. While greedy search is computationally efficient, it may not always yield the best possible translations given the limited searching space. (2) **Beam search**, as illustrated in Figure 1.3, considers multiple potential translations at each step by keeping track of a fixed number of the best partial translations named beams. Let B be the beam size. At each step, it expands each of the B sequences by considering all possible next subwords. For each partial sequence, it calculates the total score (the sum of log probabilities) for each possible extension and keeps only the B sequences with the highest total scores. Formally, given a partial sequence $\boldsymbol{y}_{1:t-1}$ and its score $S(\boldsymbol{y}_{1:t-1})$, the score for an extended sequence $\boldsymbol{y}_{1:t} = (\boldsymbol{y}_{1:t-1}, a)$ with a new subword a is:

$$S(\boldsymbol{y}_{1:t}) = S(\boldsymbol{y}_{1:t-1}) + \log P(a|\boldsymbol{y}_{1:t-1}) \tag{1.5}$$

Beam search continues until each beam produces an end-of-sequence token or reaches the maximum sequence length. The final output is the sequence among the B candidates with the highest overall score. This algorithm enlarges the searching space and usually can improve the translation quality compared to greedy search.

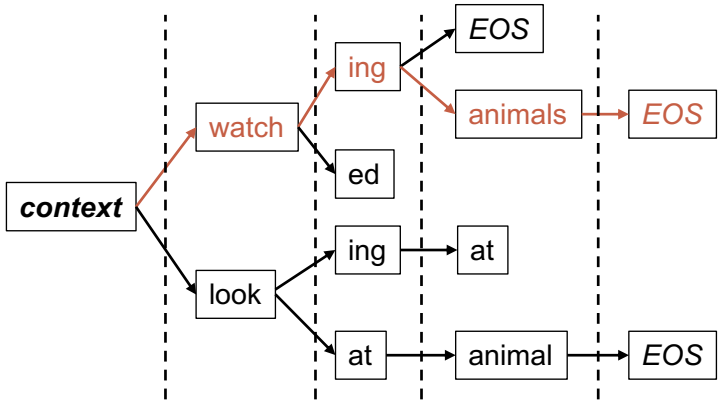


Figure 1.3: Beam search decoding algorithm. Context represents previously generated subwords. EOS is a special token denoting the end of generation. The dashed line shows the boundary of each step. The pink path shows the output subword sequence with the highest conditional probability.

1.4 Challenges and Proposed Methods

Despite the advantages over using words or characters, using subwords in NMT brings three new challenges in the segmentation, encoding, and decoding phases, especially in low-resource scenarios. 1) One word can be segmented into different subword segmentations. How do we determine the best one or best-k segmentation? 2) Current architecture encodes one segmentation of one word. How do we leverage multiple subword segmentations to capture the multi-perspective information of one word? 3) Current decoding algorithm generates one subword in each step, resulting in one segmentation for a word. However, there are multiple possible subword segmentations that form the same word. How do we incorporate them during decoding? The following paragraphs detail each problem and introduce how we address them.

In the subword segmentation phase, one word can be segmented into different subword segmentations, and the way of segmentation will largely affect the downstream tasks. Although for high-resource languages, there are dictionaries with stem information that can be leveraged for segmentation [7], for low-resource languages and long-tailed words of high-resource languages, such dictionaries are

scarce or nonexistent. Therefore, the first challenge is to determine the optimal subword segmentation in a data-driven way. For a deterministic segmenter, we would like to obtain the segmentation with the highest probability. One step further, we would like to calculate the probabilities of all possible segmentations $\boldsymbol{a} \in S_x$ for a stochastic segmenter. Previous frequency-based methods [116, 61] result in suboptimal segmentations, which are non-linguistically-motivated and show low translation quality, especially for low-resource scenarios. We propose two neural methods: SelfSeg which is based on self-supervised training and BERTSeg which is based on semantic embedding from the BERT model. They are trained to maximize the marginal likelihood of one word by maximizing the probabilities of all possible segmentations. Therefore, the segmenter generates segmentations with corresponding probabilities; thus, we can select the highest one or top-n candidates.

In the encoding phase, the previous method [135] usually encodes one segmentation to represent a given sentence. However, each type of subword segmenter captures a unique aspect of the input text. The byte-pair encoding (BPE) segmenter [116] based on the byte-pair encoding compression algorithm [31] represents the input text in a highly compressed token sequence [32]. The neural subword segmenter BERTSeg [121] is tuned on pre-trained BERT models and enables the use of semantic meaning and the ability to segment words into linguistically motivated subwords. However, NMT models based on a single segmenter cannot capture various aspects of the input text. This is especially essential in a low-resource scenario where we should make maximum use of the available data. To address this, we propose DiverSeg to exploit diverse segmentations from multiple subword segmenters that capture the various perspectives of each word in the encoder.

In the decoding phase, the previous greedy search or beam search [36] decoding algorithm selects one segmentation, which leads to the underutilization of knowledge learned about multiple tokenizations. As there are subword segmentations that form equivalent words, we propose the SubMerge algorithm to rescue the ignored probabilities by merging the equivalent ones. SubMerge is a nested search algorithm where the outer beam search treats the word as the min-

imal unit, and the inner beam search provides a list of word candidates and their probabilities, merging equivalent subword tokenizations. SubMerge estimates the probability of the next word more precisely, providing better guidance during inference. This algorithm is applicable to subword regularized models, which show large improvements in low-resource scenarios.

1.5 Thesis Outline

The rest of this thesis is structured as follows, shown in Figure 1.4.

In Chapter 2, we propose SelfSeg, a neural subword segmenter that yields linguistically intuitive segmentation and is faster during training and decoding compared to previous neural methods. SelfSeg takes as input a word in the form of a partially masked character sequence, optimizes the word generation probability and generates the segmentation with the maximum posterior probability, which is calculated using a dynamic programming algorithm. Additionally, we propose a regularization mechanism that allows the segmenter to generate various segmentations for one word. Moreover, it is trained in a self-supervised way that relies on only monolingual word-level data, making it applicable to low-resource languages without large-scale parallel resources.

In Chapter 3, we propose a BERT-based subword segmenter that generates subword segmentation that utilizes the contextualized semantic embeddings of words from the BERT model. During training, it maximizes the marginal probability from all possible segmentations of one word using a dynamic programming algorithm. During inference, it selects the one with the highest probability. Furthermore, we propose a probability-based regularization method that enables the segmenter to produce multiple segmentations for one word to improve the robustness of NMT systems. Based on a pre-trained BERT encoder, it only requires little training data to achieve reasonable segmentations, making it especially applicable in low-resource scenarios.

In Chapter 4, we propose DiverSeg to exploit diverse segmentations from multiple subword segmenters that capture the various perspectives of each word in the encoder. In DiverSeg, multiple segmentations are encoded using a subword

lattice input, a subword-relation-aware attention mechanism integrates relations among subwords, and a cross-granularity embedding alignment objective enhances the similarity across different segmentations of a word. We found incorporating information from multiple aspects enhances the performance of NMT, especially in low-resource scenarios.

In Chapter 5, we propose SubMerge, a decoding algorithm that merges the probabilities of multiple subword segmentations that form the same word, which we call equivalent segmentations. This is specially designed for the subword regularized NMT model. It leverages multiple subword segmentations of one target sentence during training as a data augmentation method, which is effective for low-resource scenarios. SubMerge is a nested search algorithm where the outer beam search treats the word as the minimal unit, and the inner beam search provides a list of word candidates and their probabilities, merging subword segmentations that form the equivalent word. It estimates the probability of the next word more precisely, providing better guidance during inference. We show it consistently outperforms the beam search algorithm in several low-resource MT datasets in terms of BLEU scores.

The methods proposed in this thesis have effectively addressed the issues in the subword-based NMT systems and significantly enhanced the translation performance, especially in low-resource scenarios. In Chapter 6, we summarize the thesis and outline the possible directions for future work.

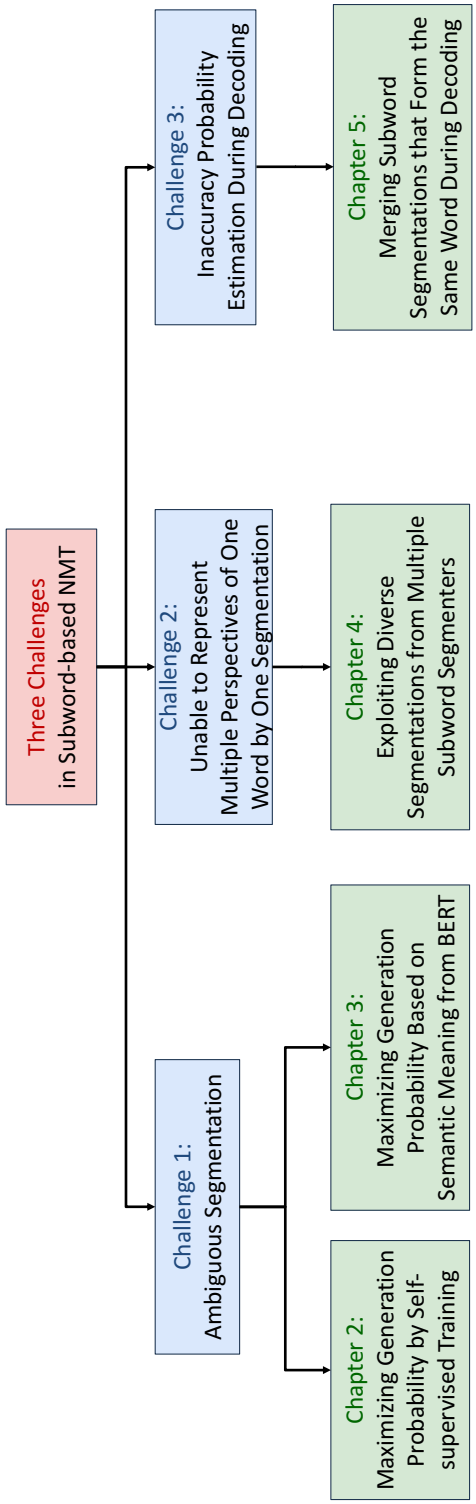


Figure 1.4: Thesis outline.

Chapter 2

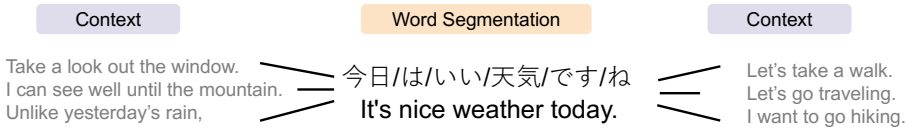
SelfSeg: A Self-supervised Subword Segmentation Method for Low-Resource Neural Machine Translation

NMT is the most prevalent approach for machine translation [126, 4, 135, 33] due to its end-to-end nature and its ability to achieve state-of-the-art translations [1]. Early NMT methods consider words as the minimal input unit and use a vocabulary to hold frequent words [126, 4, 73, 50]. However, they face the out-of-vocabulary (OOV) problem due to the limited size of the vocabulary and the unlimited variety of words in the test data. Even with a very large vocabulary that covers most words in the train set, for morphologically rich languages such as German, there are still 3% of new types of words that appear in the test set [49]. This largely hinders the translation quality of sentences with many rare words [126, 4].

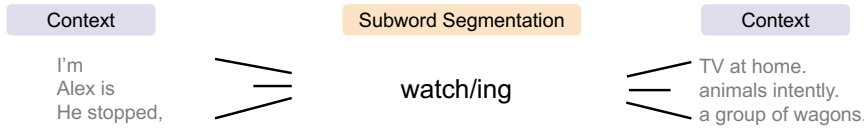
Subword segmentation is dedicated to addressing the OOV problem by segmenting rare words into subwords or characters that are present in a vocabulary. Frequency-based methods first use a monolingual corpus to build a subword vocabulary that contains characters, high-frequency subword fragments and com-

mon words. During decoding, for each word or sentence, it recursively combines an adjacent fragment pair that occurs most frequently according to the subword vocabulary, starting from characters [116, 62]. The main limitation is that these segmentation methods are not optimized for downstream tasks, such as NMT. DPE [41], a recently proposed neural subword segmentation approach, views the target sentence as a latent variable whose probability is the sum of the probability of all possible segmentations. The probability of each segmentation is calculated by a transformer model conditioned on the source sentence. It optimizes the target sentence probability in the training phase and outputs the segmentation with maximum posterior probability in the decoding phase. The DPE work also shows the importance of optimizing subword segmentation for the MT task. Different from BPE [116], it uses parallel data and deploys a neural sequence-to-sequence model for the segmentation. This is a double-edged sword: on one hand, using a sequence-to-sequence neural model enables the segmentation to be aware of all past tokens where BPE does not. Because the NMT decoder is also aware of all past tokens, this segmentation approach may be optimal; on the other hand, it is not practical neither in low-resource scenarios where large parallel corpora are not available, nor in high-resource scenarios where training and decoding take hours to days.

Leveraging existing large-scale monolingual data through self-supervised learning methods significantly reduces the need for parallel corpora, which is essential for low-resource scenarios. Predicting masked tokens is a promising task to provide training signals for an encoder that could be fine-tuned for a variety of downstream tasks [24, 78], or an encoder-decoder model which could boost the MT tasks [123]. Although relying on monolingual data obviates the need for parallel corpora, the DPE method will still be slow as entire sequences have to be processed. In order to speed up the model, we propose that words be used instead of sentences. The motivation comes from the examples in Figures 2.1(a) and 2.1(b). In Figure 2.1(a), for a Japanese word segmentation task, the sentence will be consistently segmented in different document-level contexts. It is similar for subword segmentation as presented in Figure 2.1(b), where we don't need sentence-level information. For example, the word "watching" should be consistently segmented



(a) Japanese word segmentation task. The sentence is consistently segmented with different document-level contexts. For the context we show the translated English references only.

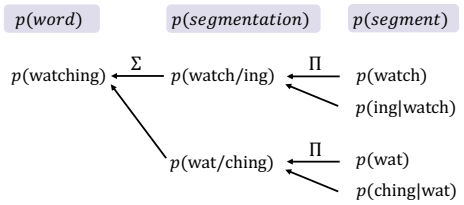


(b) English subword segmentation task. The word is consistently segmented with different sentence-level contexts.

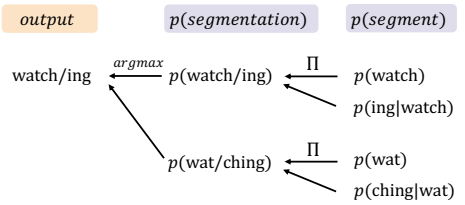
Figure 2.1: Segmentation is a self-contained task where context information is not required.

into "watch+ing" no matter which sentence the word is in. This insight can help us go from sentence-level data to word-level data to train the subword segmenter, which significantly improves the training and decoding speed, because the training requires only word-level data and one type of word needs to be decoded only once.

Based on these observations, we propose SelfSeg, a subword segmenter that trained on monolingual word-level data. It uses a neural model to optimize the word generation probability conditioned on partially masked words, and outputs the segmentation with the maximum posterior probability. The decoding is fast because it only needs to decode each unique word once. To speed up the training phase, we propose a word frequency normalization method that adjusts the frequencies for frequent and rare words. Furthermore, motivated by Provilkov et al. [106] we also implement a regularization method on top of SelfSeg which provides multiple segmentations of the same word. We conduct experiments for low-, middle- and high-resource language pairs using the corpora from Asian Language Treebank (ALT), IWSLT and WMT. We show that SelfSeg yields segmentations that achieve better translation quality of up to 1.1-1.3 BLEU compared to exist-



(a) During the training phase model maximizes the probability of one word.



(b) During the decoding phase model seeks the segmentation with the highest probability.

Figure 2.2: The training and decoding steps for subword segmentation.

ing approaches such as BPE [116], SentencePiece [61], DPE [41] and VOLT [150]. Additionally, we show that in low-resource settings regularized SelfSeg not only outperforms BPE by 4.3 BLEU but also BPE-dropout [106] by 1.2 BLEU. We also provide analyses exploring various aspects of SelfSeg. Our contributions are as follows:

- **We propose SelfSeg**, a neural subword segmentation method that relies on only monolingual word-level data with masking strategies, together with word-frequency normalization strategies to speed up the training, and a regularization mechanism.
- **Experimental results show significant BLEU score improvements** over existing works in low-resource scenarios, as well as a significant increase in training and decoding speed compared to neural approaches such as DPE.
- **We provide extensive analysis**, including the effect of different masking methods and normalization methods, and why monolingual word-level data is enough to train SelfSeg.

2.1 Related Work

In this section, we introduce two categories of subword segmentation methods, namely, non-neural and neural methods. In addition, we introduce the prevalent self-supervised learning paradigm.

2.1.1 Non-Neural Subword Segmentation

Initial works on NMT used word-level vocabularies that could only represent the most frequent words, leading to the OOV problem [126]. Character-based or byte-based approaches solve the OOV problem, however, they introduce higher computational complexity and thus translation latency because they generate longer sequences and require deeper-stacked models (often equipped with pre-layer normalization) [39, 53, 20, 69, 72, 16, 117]. Fully character-based NMT systems show higher translation quality compared with word-based systems, especially for morphologically rich languages [53, 20, 69], while a hybrid word-character model shows a larger improvement [72]. A recent study further represents every computerized text as a sequence of bytes via UTF-8 [117].

Subword segmentation methods address both the OOV problem and the computational cost of the character-based methods, thus becoming an indispensable pre-processing step for modern NMT models [116, 106, 142, 112, 62, 61]. Sennrich et al. [116] adapt BPE compression algorithm [31] to the task of subword segmentation (in this paper, we use the name BPE to refer specifically to BPE for subword segmentation). BPE detects repeated patterns in the text and compresses them into one subword. Specifically, it initializes a vocabulary of all types of characters in the training corpora, and adds frequent fragments and words into it. During decoding, a greedy algorithm recursively combines the most frequent adjacent fragment pair in the vocabulary, starting from words that are split into characters. Although not linguistically motivated, the effectiveness may come from the ability of generating shorter sequences [32]. There are several variants of the BPE method, BPE-dropout [106] is a stochastic or regularized version of BPE where words can be segmented in different ways causing a sentence to have multiple-segmented forms leading to a robust translation model. Subword regularization [60] is a regularized version of SentencePiece [61] based on a non-neural network unigram language model. VOLT [150] finds the best BPE token dictionary with a proper size. Byte-level BPE (BBPE) [142] uses bytes as the minimal unit, thus generating a compact vocabulary. WordPiece (WPM) [112] is similar to BPE where it chooses the adjacent fragment pair that maximizes the likelihood of the training data rather than based on word frequency. Different from

BPE which treats space as a special token and thus needs a tokenizer for data in different languages, SentencePiece (SPM) [62] is a language-independent method that treats the input as a raw input stream where space is not a special token. SentencePiece regularization [61] is the stochastic version of SPM where it draws multiple segmentations from one sentence to improve the robustness of the model.

The frequency-based methods however are not linguistically motivated, for example, the word "moments" will be segmented as "mom+ents" rather than "moment+s". Attempts to use a morphological analyzer for subword segmentation cannot achieve consistently translation quality improvements [154, 47]. Furthermore, this method cannot be applied to low-resource languages which lack high-quality morphological analyzers. A recent survey [90] also covers other non-neural methods such as language-specific methods [57], bayesian language models [130], and marginalization over multiple possible segmentations [15].

2.1.2 Neural Subword Segmentation

Frequency-based methods, such as BPE and SPM, are simple forms of data compression [31] to reduce entropy, which makes the corpus easy to learn and predict [2]. While we can optimize the choice of vocabulary to further reduce the entropy [150], it is more straightforward to find the segmentation that directly reduces the entropy of a neural model.

Segmentations can be optimized for a neural model to learn and generate by the *sequence modeling via segmentations* method [143]. In the training phase, it optimizes the sequence generation probability calculated by the sum of probabilities of all its possible segmentations. In the decoding phase, the segmentation with maximum a posteriori (MAP) is considered the optimal segmentation for each sentence. The sequence modeling via segmentation idea is applied to multiple NLP tasks including word segmentation [52, 125, 28], language modeling [35], NMT [59], and speech recognition [143]. During the inference of the language model, utilizing the marginal likelihood with multiple segmentations shows more robust results than one-best-segmentation [11]. DPE [41] method has applied this sequence modeling and optimization idea to the subword segmentation task. They proposed a mixed character-subword transformer and apply the dynamic

programming (DP) algorithm to accelerate the calculation of sequence modeling. However, segmentation is performed at the sentence-level and conditioned on a sentence in another language. DPE's parallel corpus requirement makes it unattractive, especially in low-resource settings [119], which motivated us to rely only on monolingual corpora. However, the mixed character-subword transformer is indispensable to our method.

2.1.3 Self-supervised Machine Learning

Self-supervised methods are becoming popular in machine learning. The advantage of this approach is that it requires only unlabeled (and often monolingual) data, which exists in large quantities. In the NLP field, using monolingual data with denoising objectives has led to significant performance gains in multiple tasks including NMT, question answering (QA) and Multi-Genre Natural Language Inference (MultiNLI) tasks [24, 123, 108, 9, 70, 71, 65, 120, 140, 82, 83, 84]. However, to the best of our knowledge, this approach has not been seriously applied to the subword segmentation task yet. Furthermore, the self-supervised method is prevalent in the field of computer vision. There are many works that use unlabeled images to pre-train models [42, 138, 103, 26, 153, 92, 145, 139, 37].

2.2 Methods

We first describe the sequence modeling via segmentation for the subword segmentation task as background in Section 2.2.1. We then describe the proposed segmenter with several masking strategies in Section 2.2.2, word frequency normalization strategies to accelerate the training speed in Section 2.2.3, and a regularization mechanism to increase the variety of the generated subwords in Section 2.2.4.

2.2.1 Background: Word Modeling via Subword Segmentations

This section describes the word modeling via subword segmentation, which is the theoretical foundation of the proposed method.

Let $\boldsymbol{x}_{1:T}$ denote a word that comprises T characters, that is $\boldsymbol{x}_{1:T} = (x_1, ..., x_T)$. Let $\boldsymbol{a}_{1:\tau_a}$ denote one segmentation of $\boldsymbol{x}_{1:T}$ that comprises τ_a subwords, that is $\boldsymbol{a}_{1:\tau_a} = (a_1, ..., a_{\tau_a})$. For each subword (or segment) a_i in a segmentation $\boldsymbol{a}_{1:\tau_a}$, it is non-empty substrings of $\boldsymbol{x}_{1:T}$ and in a predefined finite size subword vocabulary V, that is $a_i \in V$. The set of all valid segmentations for a word is represented as S_x, where $\forall \boldsymbol{a}, \boldsymbol{a} \in S_x$. Because the subword segmentation of one word is not known in advance, the probability of generating one word $\boldsymbol{x}_{1:T}$ can be defined as the sum of the probability from all subword segmentations in S_x:

$$\begin{aligned} p(\boldsymbol{x}_{1:T}) &= \sum_{P_\theta(Y|X) \neq P_\theta(\boldsymbol{y}|\boldsymbol{x}) \boldsymbol{a}_{1:\tau_a} \in S_{\boldsymbol{x}}} p(\boldsymbol{a}_{1:\tau_a}) \\ &= \sum_{\boldsymbol{a}_{1:\tau_a} \in S_{\boldsymbol{x}}} \prod_{i=1}^{\tau_a} p(a_i | a_1, ..., a_{i-1}), \end{aligned} \tag{2.1}$$

where $p(\boldsymbol{x}_{1:T})$ is the probability of the word, $p(\boldsymbol{a}_{1:\tau_a})$ is the probability of one segmentation and $p(a_i|a_1, ..., a_{i-1})$ is the probability of one segment in the segmentation $\boldsymbol{a}_{1:\tau_a}$, conditioned on previous segments, which is calculated using neural networks such as RNN or Transformer models.

However, for a sequence of length T, there are approximately 2^T types of segmentations. Without using approximation algorithms the time complexity of calculating Eq. (2.1) will be exponential ($O(2^T)$), which makes the algorithm too slow thus impractical. To address this, we adopt the mixed character-subword transformer model [41] which takes characters as input and generates subwords as output. The model represent the history information by prefix characters $x_1, ..., x_j$ instead of subwords $a_1, ..., a_{i-1}$, where $j = index(a_i) - 1$. Therefore, we have an approximate word probability:

$$p(\boldsymbol{x}_{1:T}) = \sum_{\boldsymbol{a}_{1:\tau_a} \in S_{\boldsymbol{x}}} \prod_{i=1}^{\tau_a} p(a_i | x_1, ..., x_j) \tag{2.2}$$

In this way, we can calculate the word probability in the time complexity $O(T^2)$, because there are only T types prefixes as history states, from x_1, x_1x_2 to $x_1...x_T$, and only maximum T types of possible next segments from x_i, x_ix_{i+1} to $x_i...x_T$, suppose the current index is i. This is a DP algorithm and helps speed up the segmentation process.

In the training phase, the generation probability of the model for the unsegmented sequences is optimized. Figure 2.2 provides an example. During the training phase, we can obtain the probability of the word "watching" by summing the probabilities of all possible subword segmentations such as "watch+ing" and "wat+ching," where the probability of each segmentation is the product of the probability of all its segments following the chain rule, calculated by a neural model. The training objective for this unsupervised task is to maximize the generation probability of all words: $\sum_{\boldsymbol{x}_{1:T} \in D} \log P(\boldsymbol{x}_{1:T})$ where D is the training corpus consisting of the words. For one word x the marginalization $P(\boldsymbol{x}_{1:T})$ is the sum of probabilities of all possible segmentations, calculated through Eq. (2.2). The gradient is calculated automatically through PyTorch and then propagated. The detailed calculation process can be found in Section 3.1 of the sequence modeling work [143]. In the decoding phase, we calculate the probabilities of all segmentations and then trace the one with maximum probability as the optimal segmentation.

2.2.2 SelfSeg: Self-supervised Subword Segmentation Method

We propose a self-supervised method to train a subword segmenter. Given a masked version of one word, the segmenter maximizes the likelihood of all segmentations of the word during training, and selects one segmentation with the highest likelihood during decoding.

The masked version of the word is denoted by $\boldsymbol{x}_M$. And we maximize the generation probability of word $\boldsymbol{x}_{1:T}$ during training by the following objective:

$$\log p(\boldsymbol{x}_{1:T}|\boldsymbol{x}_M) = \log \sum_{\boldsymbol{a}_{1:\tau_a} \in S_{\boldsymbol{x}}} \prod_{i=1}^{\tau_a} p(a_i|\boldsymbol{x}_M, x_1, ..., x_j) \tag{2.3}$$

We propose the **charMASS** to generate $\boldsymbol{x}_M$:

- **charMASS**: character-level MAsked Sequence-to-Sequence pre-training (charMASS), where half of the consecutive characters in one word are masked. We select the start position of the span from the indexes of the first half of the characters.

In addition, we consider three alternatives:

- **subwordMASS**: subword level MAsked Sequence-to-Sequence pre-training (MASS), where half of the consecutive subword segments in one word are masked. We select the start position of the span from the indexes of the first half of the subwords.

- **subwordMASK**: strategy used in the MASKed language model, where every subword segment is individually masked with a certain probability. We set it to 15% following the BERT paper [24].

- **w/o masking**: where we set $\boldsymbol{x}_M$ to the original word $\boldsymbol{x}$ without any masks.

Figure 2.3 illustrates the charMASS method. We directly mask characters in charMASS. However, we generate an initial segmentation using existing subword segmentation methods such as BPE [116], and mask part of the subwords. We generate the next subword possibilities for each position. The training objective is to maximize the possibility of all paths and in the decoding phase we retrace the optimal path. We create the word-level data by splitting sentence-level data into one word per line format. During decoding, we decode each type of word once which accelerates the decoding phase.

2.2.3 Word Frequency Normalization

We propose frequency normalization methods to speed up the training phase. The motivation is the observation that high-frequency words make up a large part of the training set, such as the words “the” and “is”. However, they can not provide sufficient training signals because most of them are short and non-compound words and tend to stay unsegmented.

Suppose word w_i occures q_i times in the corpus. And $freq$ is a function that maps w_i into q_i. We propose normalizing function $norm$ acting on the function $freq$ and generate normalized frequency nq_i for each word, that is $norm \circ freq(w_i) = nq_i$.

We propose the **Threshold** as $norm$

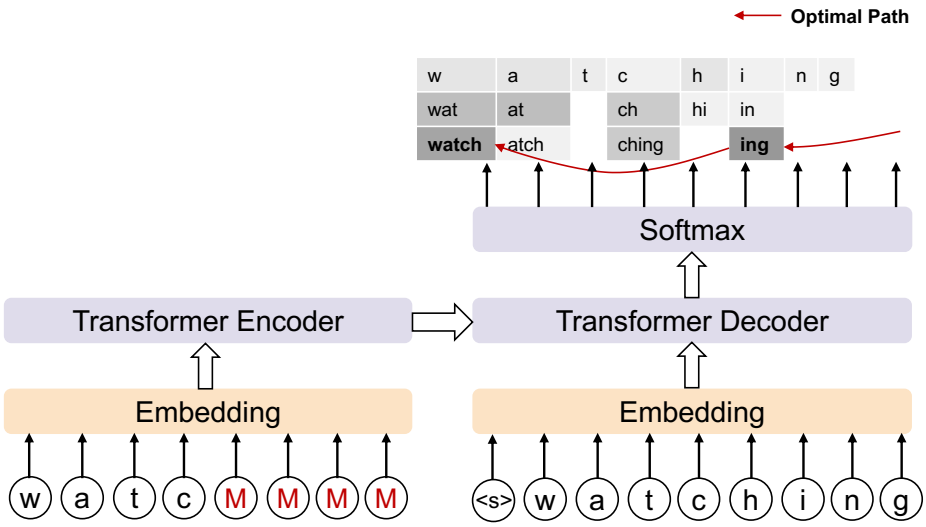


Figure 2.3: **Mixed character-subword transformer.** The input of the encoder is one word with masks. The output of the decoder is the possibilities of the next subwords in each position. We optimize all paths during training and retrace the optimal path during decoding.

- **Threshold**: $threshold(x) = \lfloor x/d \rfloor$, where we remove words with frequency lower than a threshold d and reduce the frequency for other words. We set d to 10.

In addition, we consider three alternatives:

- **Sqrt**: $sqrt(x) = \lfloor \sqrt{x} \rfloor$, in this way we reserve all types of words while especially reduce the frequency of high-frequency words.

- **Log**: $log(x) = \lfloor \log_2 x \rfloor$, where we also reserve all types of words and cuts the frequency of high-frequency words more strongly.

- **One**: $One(x) = 1$, where we retain only the type information and removes the frequency information.

We create the training data by 1) obtaining a word-frequency table from the corpus, 2) applying the normalizing function and obtaining the normalized word-

frequency table, and 3) copying each word w_i by nq_i times and then shuffle the dataset.

2.2.4 SelfSeg Regularization

Algorithm 1 shows the proposed SelfSeg-Regularization algorithm that is to increase the variety of the generated subwords. At each position i of word $\boldsymbol{x}$ during decoding, we calculate the scores β_j of choosing the subword $\boldsymbol{x}_{j:i}$. Instead of selecting the index j with the highest score, we perform weighted random sampling to draw the next subword. As shown in Line 5, the weights are calculated by feeding the probability of each index j to a softmax function with temperature t to control the diversity. We save the idx_i and retrace the segmentation $\boldsymbol{z}$ for each run. During decoding, for each type of word, we run the algorithm N times to generate a list of N segmentations.

Algorithm 1: SelfSeg-Regularization

Input: $\boldsymbol{x}$ is a word containing T characters, V is a subword vocabulary, t is the temperature hyperparameter.

Output: Segmentation $\boldsymbol{z}$

1 **for** $i \leftarrow 1$ **to** T **do**
2 **for** $j \leftarrow 1$ **to** i **do**
3 **if** $\boldsymbol{x}_{j:i} \in V$ **then**
4 $\beta_j \leftarrow \alpha_{j-1} + \log P(\boldsymbol{x}_{j:i} | x_1, .., x_{j-1})$;
5 $idx_i \leftarrow randomChoice([1, ..., i], weights = softmax(\beta/t))$;
6 $\alpha_i \leftarrow \beta_{idx_i}$;
7 $\boldsymbol{z} \leftarrow retrace(\boldsymbol{idx})$;
8 **return** $\boldsymbol{z}$;

2.3 Experimental Settings

2.3.1 Datasets

We experimented with low-resource, middle-resource, and high-resource MT settings. The datasets are listed in Table 2.1, where the size of the vocabulary is set for both the segmenters[1] and NMT models for all methods, if not otherwise specified. We applied Juman++ [132] for Japanese, Stanford-tokenizer [75] for Chinese and Moses tokenizer [58] to data of all the other languages. We normalized Romanian data and removed diacritics following previous work [115].

Dataset	Train	Valid	Test	Vocab
ALT Asian Langs-En	$18k$	$1,000$	$1,018$	$8k$
IWSLT15 Vi-En	$133k$	$1,553$	$1,268$	$8k$
WMT16 Ro-En	$612k$	$1,999$	$1,999$	$8k$
WMT15 Fi-En	$1.8M$	$1,500$	$1,370$	$8k$
WMT14 De-En	$4.5M$	$45,781$	$3,003$	$8k$
WMT14 Fr-En	$10.0M$	$26,875$	$3,003$	$8k$

Table 2.1: Statistics of the corpora used in the NMT experiments.

Low-resource Setting We used the ALT multi-way parallel dataset [131]. We used English and 6 Asian languages: Filipino (Fil), Indonesian (Id), Japanese (Ja), Malay (Ms), Vietnamese (Vi), and simplified Chinese (Zh). The SelfSeg segmenter is applied to only the target language side. Therefore, we train one SelfSeg segmenter using $50,000$ randomly selected English sentences from news commentary corpus[2] for all the Asian language to English directions. We trained a Japanese SelfSeg segmenter using $98k$ Japanese sentences from KFTT dataset [98] for English to Japanese direction and an Indonesian SelfSeg segmenter using $62k$ Indonesian sentences from the Indonesian news commentary corpus for English to Indonesian direction. We trained one DPE [41] segmenter for each language pair in ALT using the corresponding $18,088$ parallel sentences. We trained BPE [116],

[1] We keep in line with SPM's definition of vocabulary size.

[2] `http://data.statmt.org/news-commentary/v14/`

BPE-dropout [106] and VOLT [150] segmenters using the 18,088 monolingual sentences in ALT for the corresponding languages.

Middle- and High- Resource Setting We used the IWSLT'15 Vietnamese-English, WMT'16 Romanian-English, WMT'15 Finnish-English, WMT'14 German-English,[3] and WMT'14 French-English[4] corpora. We use the first 10.0 million parallel sentence pairs in the WMT'14 French-English train set in our experiments. We used English monolingual sentences from the training set of each corpus as the training data for all methods except DPE. For the DPE method, we used the parallel sentences from the train sets following the official implementation,[5] where the input of the encoder is the sentence in the source language, and the predicted output is the sentence in the target language.

2.3.2 Segmenter Model Settings

BPE, SentencePiece, VOLT, and BPE-dropout For BPE [116], we used a widely adopted toolkit[6] with model type as BPE. For SentencePiece, we use unigram language model implemented in the toolkit. For VOLT [150], we used the default setting in the official implementation.[7] For BPE-dropout [106], we apply dynamic dropout for each epoch and with a drop rate of 0.1 (0.05 for English→Japanese) selected by hyperparameter tunning.

SelfSeg and DPE For SelfSeg, we used charMASS as the masking strategy and Threshold as the word frequency normalization strategy in Section 2.4. Detailed analysis of the masking strategies and frequency normalization strategies are shown in Section 2.5. For the SelfSeg and DPE, we used the mixed character-subword transformer model with DP algorithm, where the transformer architecture is of 4 encoder layers and 4 decoder layers, dropout of 0.3, inverse sqrt learning rate scheduler with 4,000 warmup steps, and the dynamic programming

[3] https://github.com/facebookresearch/fairseq/blob/main/examples/translation/prepare-wmt14en2de.sh

[4] https://github.com/facebookresearch/fairseq/blob/main/examples/translation/prepare-wmt14en2fr.sh

[5] https://github.com/xlhex/dpe

[6] https://github.com/google/sentencepiece

[7] https://github.com/Jingjing-NLP/VOLT

cross-entropy criterion as described in the DPE method. We set the number of training epochs to 50, which is large enough for convergence. Additionally, for the mixed character-subword transformer model, the vocabulary V should contain all characters to prevent OOV problems and commonly used subwords. Here we used a subword vocabulary generated by the BPE algorithm [116], which satisfies the two conditions, following previous work [41].

SelfSeg-Regularization We set N to 10 and t to 10 (t to 3 for English to Japanese) in Algorithm 1. In the MT experiments, at each epoch, we dynamically generate a segmentation for each sentence in the dataset. For each word in the sentence, we randomly select one of the N segmentations.

Note that DPE, SelfSeg, and SelfSeg-Regularization are used to segment only the target side in the MT experiments. The source-side simply uses BPE data for SelfSeg and BPE-dropout data for the SelfSeg-Regularization. This is because the loss function of the segmenter is to maximize the generation probability. Therefore, these segmentations are effective for the target sentence. This is also studied in the DPE work [41].

2.3.3 NMT Settings

We used the fairseq framework [100] with the Transformer [135] architecture with 6 layer encoder (except for Filipino where 4 encoder layers were sufficient), 6 layer decoder and 1 attention head, decided through hyperparameter tuning as suggested by Rubino et al. [110]. Dropout of 0.1 and label smoothing of 0.1 is used. We used layer normalization [64] for both the encoder and decoder. We used a vocabulary size of $8,000$ for the NMT models. Batch-size is set to $1,024$ tokens. We used the ADAM optimizer [55] with betas (0.9, 0.98), warm-up of $4,000$ steps followed by decay, and performed early stopping based on the validation set BLEU. We used a beam size of 12 and a length penalty of 1.4 for decoding. We reported sacreBLEU [105], METEOR [6], and BLEURT [113] on detokenized outputs.

	Fil→En	Id→En	Ja→En	Ms→En	Vi→En	Zh→En	Avg	Δ
w/o Regularization								
BPE [116]	23.09	25.70	9.42	28.19	19.94	12.21	19.76	0.00
SentencePiece [61]	23.71	25.49	9.94	27.72	18.58	11.74	19.53	−0.23
VOLT [150]	22.99	25.05	10.56	27.91	21.64	11.31	19.91	0.15
DPE [41]	24.04	26.66	9.93	27.89	20.06	10.72	19.88	0.13
SelfSeg	25.20*	27.10*	11.39*	28.15	22.44*	12.03	21.05	1.29
With Regularization								
BPE-dropout [106]	28.18	28.02	12.84	31.59	23.67	**13.91**	23.04	3.13
SelfSeg-Regularization	**29.94***	**29.34***	**15.23***	**32.31***	**23.93***	13.64*	**24.07**	**4.31**

Table 2.2: **Low-resource Asian languages to English MT results.** The numbers in the table indicate the sacreBLEU scores. We show the average BLEU scores (Avg) and the improvements (Δ) over the BPE method. Methods are separated into without regularization and with regularization. Statistical significance [56] is indicated by * ($p < 0.001$) between the BPE baseline and the proposed methods in each direction.

2.4 Results

We report the performance of NMT as well as the training/decoding speed of our methods compared with existing works in this section.

2.4.1 MT Results

Low-Resource Scenario Tables 2.2 and 2.3 show low-resource Asian language to English NMT results. SelfSeg-Regularization achieves the highest BLEU scores among all methods in almost all directions, outperforming the BPE method by 4.31 BLEU scores on average. Among methods without regularization, proposed SelfSeg outperforms not only frequency-based methods but also neural method DPE. However, we observed that for the Ms→En and Zh→En directions, the proposed SelfSeg method is slightly worse (which is not significant) than the BPE method. In particular, we find that both neural methods (DPE and SelfSeg) perform relatively poorly in the Zh→En direction. Actually, for all directions

	Fil→En	**Id→En**	**Ja→En**	**Ms→En**	**Vi→En**	**Zh→En**	**Avg**	**Δ**
w/o Regularization								
BPE [116]	29.1/45.0	31.1/49.2	20.1/32.4	32.7/52.0	27.6/44.6	22.9/36.9	27.2/43.3	0.0/0.0
SentencePiece [61]	29.7/46.1	31.2/48.9	21.0/33.8	32.2/51.0	26.6/42.4	21.6/34.2	27.0/42.7	-0.2/-0.6
VOLT [150]	29.2/45.2	31.0/48.8	21.2/34.2	32.5/51.1	28.4/46.6	22.2/35.5	27.4/43.6	0.2/0.2
DPE [41]	29.7/46.5	31.8/50.5	21.1/34.4	32.5/51.6	26.9/43.9	21.5/35.3	27.3/43.7	0.0/0.3
SelfSeg	30.2/47.3	32.0/51.3	21.5/35.3	32.6/52.3	28.4/46.3	22.4/36.5	27.9/44.8	0.6/1.5
With Regularization								
BPE-dropout [106]	32.0/51.1	33.0/52.2	22.8/36.9	34.8/55.8	29.1/48.3	**23.6**/38.8	29.2/47.2	2.0/3.8
SelfSeg-Regularization	**33.2/52.6**	**33.5/53.9**	**24.4/40.1**	**35.0/56.5**	**29.7/48.7**	23.0/**38.8**	**29.8/48.4**	**2.6/5.1**

Table 2.3: **Low-resource Asian languages to English MT results.** The numbers in the table indicate the METEOR [6]/BLEURT [113] scores. We show the average scores and the improvements (Δ) over the BPE method.

SelfSeg are better (or worse) than BPE, DPE is also better (or worse) than BPE. Therefore, we assume that for segmentations generated by neural segmenters, the performance does have a correlation with the source language. We will leave the in-depth exploration of this question as future work. We found that adding regularization yields significant BLEU score improvement in the low-resource situation. The SelfSeg-Regularization method substantially improves over BPE. Results of the METEOR and BLEURT evaluation metrics also show similar trends.

Tables 2.4 and 2.5 show English to Japanese and Indonesian NMT results of the ALT dataset, English to Romanian results of the WMT16 Ro-En dataset, and the English to Finnish results of the WMT15 Fi-En dataset. In English to Indonesian direction, the SelfSeg-Regularization outperforms all baseline methods substantially. For the English to Japanese direction, the improvement is limited because the average length of the Japanese words in the ALT dataset is short, only 1.87, resulting in less variety in word segmentation. As a comparison, the average length of English words is 4.54 and the average length of Indonesian words is 5.50. This may explain why regularization brings more improvement for English→Indonesian than English→Japanese. For the En→Ro and En→Fi translation directions, we observed that the SelfSeg performs best among the w/o regu-

	En→Ja	**En→Id**	**En→Ro**	**En→Fi**
w/o Regularization				
BPE [116]	12.69	28.08	33.62	15.54
SentencePiece [61]	12.58	26.01	33.17	15.75
VOLT [150]	13.11	28.46	33.13	15.24
DPE [41]	13.46	29.29	33.71	15.27
SelfSeg	13.26	29.00	33.72	15.85
With Regularization				
BPE-dropout [106]	**14.97**	30.74	**35.48**	**17.04**
SelfSeg-Regularization	14.31*	**33.77***	35.47*	16.93*

Table 2.4: En→Other language results. En→Ja and En→Id directions are from the ALT dataset, En→Ro direction is from the WMT16 Ro-En dataset and En→Fi direction is from the WMT15 Fi-En dataset. Statistical significance [56] is indicated by * ($p < 0.001$) between the BPE baseline and the proposed methods in each direction.

larization methods whereas the results of BPE-dropout and SelfSeg-regularization are comparable in terms of the BLEU, METEOR and BLEURT metrics.

Middle- and High-Resource Scenario The results for the middle- and high-resource scenarios are presented in Tables 2.6 and 2.7. The proposed methods show up to 1.9 BLEU score improvement, 1.0 METEOR score improvement and 1.4 BLEURT score improvement compared with BPE and outperform other baseline methods for all datasets except the high-resource settings WMT14 De→En and WMT14 Fr→En. Additionally, the neural methods (DPE and SelfSeg) outperform non-neural methods (BPE and SentencePiece) in most settings.

We find that the effect of subword segmentation on performance becomes marginal as the training data becomes larger. For the WMT14 De→En and WMT14 Fr→En directions, we found no improvement over BPE. Additionally, two methods with regularization didn't show better results than methods without

	En→Ja	**En→Id**	**En→Ro**	**En→Fi**
w/o Regularization				
BPE [116]	24.87/17.94	30.13/46.66	31.19/66.92	19.84/62.02
SentencePiece [61]	23.91/17.50	29.29/45.99	31.20/66.38	20.31/63.16
VOLT [150]	25.10/18.45	30.31/46.70	31.12/66.02	19.79/61.86
DPE [41]	25.24/18.39	30.76/48.00	31.42/66.69	19.94/62.86
SelfSeg	25.24/18.45	30.59/47.97	31.08/67.30	19.98/62.71
With Regularization				
BPE-dropout [106]	**26.00/20.84**	31.56/48.66	**32.10/69.59**	20.90/65.20
SelfSeg-Regularization	25.47/20.65	**33.07/50.72**	32.04/69.44	**21.05/65.61**

Table 2.5: En→Other language results. En→Ja and En→Id directions are from the ALT dataset, En→Ro direction is from the WMT16 Ro-En dataset and En→Fi direction is from the WMT15 Fi-En dataset. The numbers in the table indicate the METEOR [6]/BLEURT [113] scores.

regularization. This is also shown in the DPE work [41] where the improvement is marginal, and the BPE-dropout work [106] where the dropout hurts the performance for larger datasets. Therefore, one of the limitations of our approach is the small to medium sized MT dataset. Note that we didn't conduct DPE experiments on the WMT14 De→En and WMT14 Fr→En datasets because of excessive computational resource consumption as shown in Section 2.4.2.

2.4.2 Training and Decoding Speeds

Figure 2.4 provides the training speeds and decoding speeds of SelfSeg, BPE and DPE. The training speed of SelfSeg is 17.8x faster than the DPE method on the WMT'16 Ro-En dataset and 18.7x faster on the ALT dataset. Although the speed is not as fast as the BPE method, the training process can finish in approximately one hour for a $612k$ size dataset, which is much more acceptable than the DPE method which requires more than one day.

	IWSLT15 Vi→En	WMT16 Ro→En	WMT15 Fi→En	WMT14 De→En	WMT14 Fr→En
w/o Regularization					
BPE [116]	27.09	32.54	17.45	31.00	34.97
SentencePiece [61]	26.58	31.48	17.74	30.62	34.92
VOLT [150]	27.16	31.89	17.25	**31.24**	**35.60**
DPE [41]	27.40	33.05	17.51	-	-
SelfSeg	28.19	32.59	18.00	30.82	34.91
With Regularization					
BPE-dropout [106]	28.76	33.59	18.89	30.56	34.38
SelfSeg-Regularization	**29.01***	**34.01***	**19.01***	30.59	34.39

Table 2.6: Middle- and high-resource MT results with BLEU scores. Statistical significance [56] is indicated by * ($p < 0.001$) between the BPE baseline and the proposed methods in each direction.

	IWSLT15 Vi→En	WMT16 Ro→En	WMT15 Fi→En	WMT14 De→En	WMT14 Fr→En
w/o Regularization					
BPE [116]	31.16/57.75	35.18/61.99	27.06/55.83	34.09/64.66	36.24/67.04
SentencePiece [61]	30.63/56.42	34.43/60.64	27.32/56.45	33.49/63.68	36.74/67.46
VOLT [150]	30.90/57.13	34.90/61.28	26.73/55.44	**34.04/64.60**	**37.00/67.80**
DPE [41]	31.07/57.61	35.47/62.28	27.38/55.96	-	-
SelfSeg	31.46/58.50	35.26/62.44	27.45/56.67	33.54/64.42	36.17/67.31
With Regularization					
BPE-dropout [106]	32.09/59.07	35.73/**63.38**	**28.39/58.43**	33.59/64.18	35.95/66.88
SelfSeg-Regularization	**32.15/59.17**	**35.84**/63.35	28.11/57.87	33.55/63.72	36.41/66.77

Table 2.7: Middle- and high-resource MT results. The numbers in the table indicate the METEOR [6]/BLEURT [113] scores.

The decoding speed of SelfSeg is 5.9x on a smaller ALT dataset and 36.8x on a larger WMT16 Ro-En dataset compared with the DPE method. This is because, according to Zipf's law, the number of distinct words in a document increases much slower compared with the increment of the total number of words in the document, i.e $\Delta O(\#distinct\ words) \ll \Delta O(\#total\ number\ of\ words)$. As shown in Table 2.8, for the smaller ALT dataset, DPE needs to decode 14.3x more tokens than SelfSeg, however, for the larger WMT'16 Ro-En dataset, DPE needs to decode 186.5x more tokens than SelfSeg. Therefore, the advantage of SelfSeg becomes greater when the corpus becomes bigger because it only needs to decode

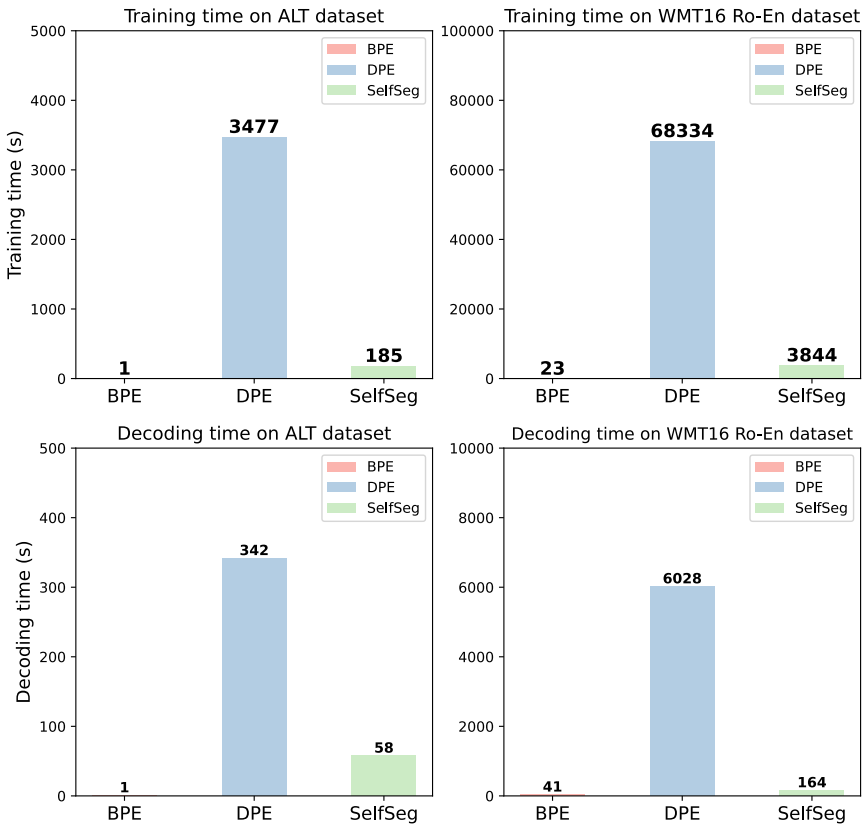


Figure 2.4: The training and decoding speeds of BPE, DPE, and SelfSeg methods on two datasets.

each distinct word once in the corpus.

The SelfSeg-Regularization method is only applied in the decoding phase, therefore the training time is the same as SelfSeg. During decoding, it generates N segmentations for one word, therefore, the time consumption is N times compared with SelfSeg. When we set N to 10, the decoding time will still be less than that of DPE.

The speed improvement is important because, in a latency-sensitive scenario, it is important to minimize as many computations as possible. Given that SelfSeg can lead to more intuitive segmentations (as seen in Section 2.5.6) and better translation than BPE while being significantly faster than DPE, which indicates that the proposed method can be very reliable in a low-latency scenario.

	ALT	WMT16 Ro-En
DPE	478k	16M
SelfSeg	33k	70k

Table 2.8: Number of tokens DPE and SelfSeg methods require to decode for each dataset.

	ALT Asian Langs→En	IWSLT15 Vi→En	WMT16 Ro→En	WMT15 Fi→En	WMT14 De→En	WMT14 Fr→En
w/o Regularization						
BPE [116]	34.04	24.80	30.40	26.63	35.41	35.33
SentencePiece [61]	41.00	28.15	35.30	29.39	35.79	35.15
VOLT [150]	34.04	25.14	29.60	26.03	32.88	32.74
DPE [41]	34.17	24.62	27.44	25.67	-	-
Selfseg	40.31	25.92	34.23	29.19	36.18	36.74
With Regularization						
BPE-dropout [106]	47.20	32.69	44.36	38.82	47.51	49.29
Selfseg-regularization	44.51	32.18	43.50	38.00	46.67	49.25

Table 2.9: The average number of subwords in each English sentence by different segmenters.

As a supplement, we provide statistics on how many subwords each sentence contains. As shown in Table 2.9, there is no significant difference in the number of subwords in the sentence using different segmentation methods. For the without regularization group, the order from high to low is SentencePiece, SelfSeg, BPE, VOLT, DPE. For the with regularization group, BPE-dropout, Selfseg-regularization. This shows that the number of subwords is not a key reason for the speed difference.

2.5 Analysis

2.5.1 Masking Strategies

Table 2.10 shows the performance of using different masking strategies. The charMASS method shows the highest performance, while the performance of subwordMASS is also higher than w/o masking, whereas subwordMASK is slightly worse than w/o masking. This is because the subwordMASK objective is not very suitable for the generation task. Second, charMASS shows higher BLEU scores than subwordMASS. This is because the number of characters in the word is more than the number of subwords. During training, charMASS can generate more variants of the masked source inputs, which provides more training signals. Furthermore, results of using 1) different masking ratios and 2) consecutive or non-consecutive masking strategies for charMASS on the Vi→En direction of the IWSLT15 dataset are shown in Figure 2.5. We mask $\lfloor ratio * \#chars \rfloor$ characters in each word. For the consecutive strategy, we choose the start point of the masking span from the possible start points randomly. For the non-consecutive strategy, we shuffle a list containing 0s with the number of masking characters and 1s with the number of non-masking characters to obtain the masking positions. We found that for both consecutive masking and non-consecutive masking methods, 0.5 is the best ratio for all settings except SelfSeg-Regularization with consecutive masking, and the performance drops if the masking ratio is very high (0.9) or very low (0.1). Additionally, there is no significant difference between using consecutive masking and non-consecutive masking strategies.

	Fil→En	**Id→En**	**Ja→En**	**Ms→En**	**Vi→En**	**Zh→En**	**Avg**
charMASS	**25.20**	**27.10**	**11.39**	28.15	**22.44**	**12.03**	**21.05**
subwordMASS	24.51	26.42	11.17	**28.60**	21.15	10.97	20.47
subwordMASK	23.79	25.35	10.26	28.34	21.37	12.00	20.19
w/o mask	24.11	26.45	10.23	28.12	21.20	11.85	20.33

Table 2.10: The BLEU scores of SelfSeg with different masking strategies.

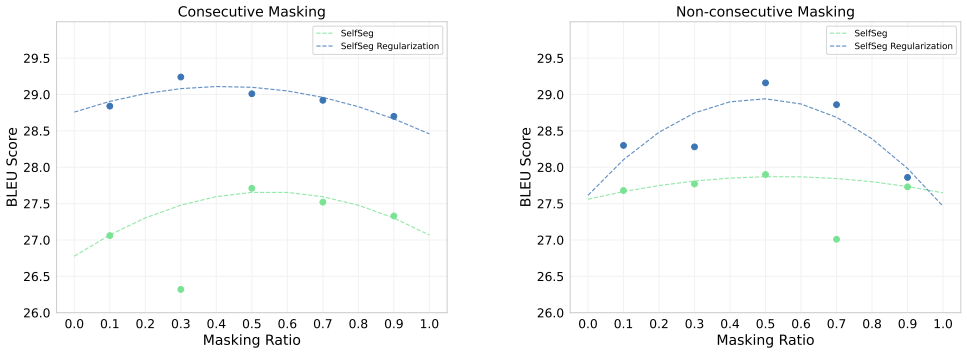


Figure 2.5: Performance of using different masking ratios and strategies for char-MASS. Left: Consecutive masking strategy. Right: Non-consecutive masking strategy. Tested on the Vi→En direction of the IWSLT15 dataset.

2.5.2 Word Frequency Normalization Strategies

Table 2.11 presents the performance of SelfSeg using different word frequency normalization strategies. We found that 1) using word frequency normalization shows comparable BLEU scores with *w/o Norm*, and 2) all strategies yield similar results except *One*, which may come from the large difference in frequency distribution between training and real data. We used subwordMASS strategy here.

	Fil→En	**Id→En**	**Ja→En**	**Ms→En**	**Vi→En**	**Zh→En**	**Avg**
w/o Norm	24.51	26.42	11.17	28.60	21.15	10.97	**20.47**
Threshold	24.74	25.86	10.42	28.82	20.90	12.06	**20.47**
Sqrt	23.48	25.95	10.64	28.31	20.64	12.04	20.18
Log	24.80	26.03	11.67	28.21	20.67	12.71	20.68
One	22.89	24.80	10.25	26.82	19.87	11.56	19.37

Table 2.11: BLEU scores of SelfSeg with different normalization strategies.

2.5.3 Types of Training Data

We demonstrate that parallel or sentence-level training data is unnecessary and monolingual word-level data is sufficient by both subword segmentation results and MT results.

Metrics The MT performance is measured by BLEU scores, and we measure the difference in subword segmentation generated by two segmenters on a given dataset through the following metric.

For each word, we define the Word Difference Rate (DIF_{word}) by Eq. (2.4), where S_1 and S_2 are sets of subword segmentations for the given $word$ generated by two segmenters. $|S|$ is the size of S, $nword$ is the frequency of $word$ in the corpus.

$$DIF_{word}(S_1, S_2, word) = \frac{\sum_{i=1}^{|S_1|} \sum_{j=1}^{|S_2|} (seg_i \neq seg_j)}{nword^2} \tag{2.4}$$

We define Corpus Different Rate (DIF_{corpus}) based on DIF_{word} in Eq. (2.5), where W is a set containing all types of words $word_i$ for the given corpus, $|W|$ is the size of W.

$$DIF_{corpus}(S_1, S_2, W) = \frac{\sum_{i=1}^{|W|} DIF_{word}(S_1, S_2, word_i) * nword_i}{\sum_{i=1}^{|W|} nword_i} \tag{2.5}$$

Additionally, if $S_1 = S_2$, DIF_{word} and DIF_{corpus} measure the consistency of segmentation of the same word in different sentences by the segmenter.

Settings We calculate DIF_{corpus} among BPE, DPE, SelfSeg-Sentence (using sentence-level data), and SelfSeg on the English part of the IWSLT'15 Vi-En dataset. All four methods use the same vocabulary. The input of SelfSeg-Sentence is a monolingual sentence instead of word during both training and decoding. subwordMASS is used for SelfSeg and SelfSeg-Sentence.

Parallel Data is Not Necessary Tables 2.12 and 2.13 present the MT results where using monolingual sentence-level data achieved higher BLEU scores than using parallel data. Figure 2.6 shows the DIF_{corpus} results. SelfSeg-Sentence gives more consistent segmentations compared with DPE (0.17% vs. 0.5%).

Sentence-level Data is Not Necessary Comparing SelfSeg-Sentence (DPE) and SelfSeg, we can find that SelfSeg using word-level data achieves higher MT

performance, showing that word-level data is enough for MT. The DPE work [41] used sentence-level data based on the assumption that one word will be segmented differently in different contexts. We found the DIF_{corpus} $(DPE, DPE, IWSLT15)$ is only 0.5%, showing that this assumption is not valid. Furthermore, we divided the words occurring in the dataset into two sets, W_{high} containing high-frequency words ($nword > 5$) and W_{low} containing low-frequency words ($nword <= 5$). We found $DIF_{corpus}(DPE, DPE, W_{high})$ is only 0.40%, on the other hand, we found $DIF_{corpus}(DPE, DPE, W_{low})$ is 6.14%. Even for the W_{low} with high DIF_{corpus}, one word should be segmented consistently. For example, DPE segments word *jumbled* into *ju+mble+d* and *j+umb+led*, and word *mended* into *me+nded* and *m+ended*, whereas the SelfSeg generates *j+umb+l+ed* and *m+end+ed*.

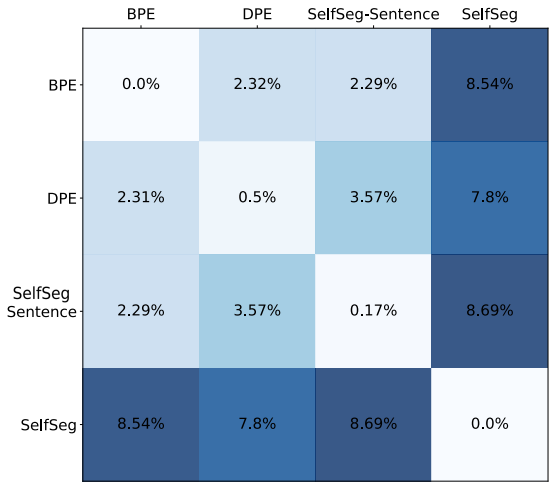


Figure 2.6: DIF_{corpus} among different segmentation methods on the IWSLT'15 Vi-En dataset.

2.5.4 Sizes of Training Data for SelfSeg

In this section, we investigate the impact of the amount of monolingual data used in the segmenter training. The results are represented in Table 2.14. The amount of English data to train the SelfSeg segmenter varies from $18k$ to $10M$, where the $18k$ setting used the ALT English data, the $50k$ and $532k$ setting used the news commentary corpus, the $4.5M$ setting used the English side of the WMT14 De-En

	Fil→En	Id→En	Ja→En	Ms→En	Vi→En	Zh→En	Avg
BPE [116]	23.09	25.70	9.42	28.19	19.94	12.21	19.76
DPE [41]	24.04	26.66	9.93	27.89	20.06	10.72	19.88
SelfSeg-Sentence	24.28	25.37	10.74	28.25	21.36	12.11	20.35
SelfSeg	24.51	26.42	11.17	28.60	21.15	10.97	**20.47**

Table 2.12: BLEU scores on the ALT dataset using segmenters trained on different types of data. DPE uses parallel sentence-level data, SelfSeg-Sentence uses monolingual sentence-level data, and SelfSeg uses monolingual word-level data.

	IWSLT15 Vi→En
BPE [116]	27.09
DPE [41]	27.40
SelfSeg-Sentence	27.79
SelfSeg	**28.19**

Table 2.13: BLEU scores on IWSLT15 Vi→En dataset with segmenters trained on different types of data.

dataset and the $10.0M$ setting used the English side of the WMT14 Fr-En dataset. We find that using more monolingual data brings performance improvement. Especially with 10M monolingual sentences from WMT14 Fr-En, the improvement reached 1.7 BLEU score compared with SelfSeg using $18k$ monolingual sentences. Although with more data, the performance of using BPE also improves, the improvement is small compared with that of SelfSeg.

2.5.5 Lightweight SelfSeg Model

We examine a lightweight segmenter model SelfSeg-Light, given that the training data is word-level and on a small scale. The architecture of SelfSeg-Light is a single-layer transformer encoder and a single-layer transformer decoder. As illustrated in Table 2.15, the performance of SelfSeg-Light is comparable with SelfSeg, which indicates that maybe there is no need to use a large model.

	Fil→En	Id→En	Ja→En	Ms→En	Vi→En	Zh→En	Avg
Size: 18k							
BPE [116]	23.09	25.70	9.42	28.19	19.94	12.21	19.76
DPE [41]	24.04	26.66	9.93	27.89	20.06	10.72	19.88
SelfSeg	24.11	25.85	11.11	28.73	20.68	10.46	**20.16**
Size: 50k							
BPE [116]	23.90	25.62	10.54	28.62	19.99	11.51	20.03
SelfSeg	24.51	26.42	11.17	28.60	21.15	10.97	**20.47**
Size: 532k							
BPE [116]	23.71	25.68	10.59	28.53	21.57	11.04	**20.19**
SelfSeg	23.96	26.40	9.93	28.01	20.92	11.66	20.15
Size: 4.5M							
BPE [116]	23.77	26.43	10.64	27.75	22.11	11.37	20.35
SelfSeg	25.60	26.45	10.09	28.57	20.23	12.22	**20.53**
Size: 10.0M							
BPE [116]	24.53	25.69	10.59	28.14	21.45	11.92	20.39
SelfSeg	26.49	27.37	11.84	29.47	23.10	12.90	**21.86**

Table 2.14: Performance of the segmenter model trained on different sizes of the training data. We only have DPE 18k because it uses ALT parallel data.

2.5.6 Segmentation Case Study

In this section, we analyze the segmentation and show why the segmentation generated by our method leads to better performance on the downstream MT task.

Table 2.16 shows examples of words with different segmentations between the BPE and SelfSeg method on the ALT dataset. We can observe that the BPE method tends to generate high-frequency subwords, due to the greedy strategy, whereas our SelfSeg, powered by the DP algorithm, tends to generate linguistically intuitive combinations of subwords for not only frequent words but also rare words. This observation is similar to that by [41]. Additionally, Table 2.17 provides some examples of subword segmentations by BPE-dropout [106] and proposed SelfSeg-Regularization. Both methods yield high diversity of segmentations while the proposed method generates more linguistically intuitive subwords.

	Fil→En	**Id→En**	**Ja→En**	**Ms→En**	**Vi→En**	**Zh→En**	**Avg**
SelfSeg	25.20	27.10	11.39	28.15	22.44	12.03	**21.05**
SelfSeg-Light	25.26	26.07	11.34	29.10	20.64	12.05	20.74

Table 2.15: BLEU scores of transformers with 4- and 1-layer. With charMASS strategy.

To verify whether our segmentation looks intuitive for the neural models, we trained neural word language models with $Transformer_{base}$ architecture,[8] the same used in the MT experiments, and checked the decoding perplexity. For each segmentation method, we train an English neural word language model on the ALT-train set and test on the ALT-test set segmented by that method. As presented in Table 2.18, the decoding perplexity of DPE and SelfSeg methods are much lower than that of the BPE method, which we assume is due to the optimization of the log marginal likelihood of the DPE method. From the results of neural LMs, we can infer that when applying our segmentations to MT tasks, the decoder tends to be more certain, as indicated by the low entropy.

2.6 Conclusion and Future Work

We proposed a novel method SelfSeg for neural subword segmentation to improve the performance of NMT and only requires monolingual word-level data. It models the word generation probability through all segmentations and chooses the segmentation with MAP. We propose masking strategies to train the model in a self-supervised manner, word-frequency normalization methods to improve the training speed, and a regularization mechanism that helps to generate segmentations with more variety. Experimental results show that NMT using proposed SelfSeg methods is either comparable to or better than NMT using BPE and DPE in low-resource to high-resource settings. And the regularization mechanism achieves a large improvement over baseline methods.

Furthermore, both the training speed and testing speed are more than ten

[8] `https://github.com/pytorch/examples/tree/master/word\_language\_model`

BPE [116]	**SelfSeg**	**BPE** [116]	**SelfSeg**
frequent words		*rare words*	
dam + aged	damage + d	d + raf + ting	d + raft + ing
com + ments	comment + s	murd + ered	murder + ed
hous + es	house + s	Net + w + orks	Net + work + s
subsequ + ently	subsequent + ly	aut + h + ored	author + ed
wat + ching	watch + ing	disag + reed	disagree + d
sec + retary	secret + ary	*one-shot words*	
un + k + n + own	un + know + n	reinfor + ces	reinforce + s
refere + es	refer + ee + s	sub + stit + utions	sub + stitution + s
langu + ages	language + s	trad + em + ar + ks	trade + mark + s
you + n + gest	young + est	ris + king	risk + ing
mom + ents	moment + s	Somet + hing	Some + thing

Table 2.16: **Examples of subword segmentations by different approaches.** The frequency of rare words are < 5, and one-shot words appear only once in the ALT training data.

times faster than those of DPE. Analyses show the context-agnostic property of the subword segmentation, therefore sentence-level training data is not required. Moreover, the segmentations given by the proposed method are more linguistically intuitive as well as easier for the neural decoder to generate as indicated by the low entropy.

Our future work will focus on several directions. First, we are implementing the pre-trained encoder such as BERT/mBERT/BART/mBART on the segmenter. The charMASS method only captures the lexical information and involving semantic information may further improve the quality. Second, we will try to extend the model to multilingual settings. In this way, we only need to train one model to pre-process data of all languages instead of training multiple

BPE-dropout [106]	**SelfSeg-Regularization**
frequent words	
sub + sequ + ently	subsequent + ly
subsequ + ently	subsequ + ent + ly
s + ub + sequ + ent + l + y	subsequent + l + y
sub + sequ + ently	subsequent + ly
subsequently	subsequ + en + t + ly
rare words	
disag + reed	disagree + d
d + is + ag + reed	disag + r + e + ed
disag + re + ed	disag + re + e +d
disag + reed	dis + ag + r + e + ed
d + is + ag + reed	disagree + d
one-shot words	
rein + for + ces	reinforce + s
re + in + f + or + ces	reinfor + ces
re + in + for + ces	reinforce + s
rein + for + ces	reinforce + s
re + in + for + ces	reinfor + c + e + s

Table 2.17: Examples of subword segmentations by BPE-dropout and SelfSeg-Regularization on the ALT dataset.

models for different languages, which can drastically reduce the training time and increase the efficiency of the application. Third, the direction of joint training of the segmenter and the downstream tasks model is also promising, where the segmenter will be aware of the downstream tasks explicitly and be optimized to improve the performance of downstream tasks. Finally, optimizing the vocabulary for sequence generation is necessary. Although the segmentations are optimized

	PPL per line	PPL per token	# tokens per line
BPE [116]	29,688.2	799.4	37.1
DPE [41]	28,498.7	816.1	34.9
SelfSeg	28,714.0	772.2	37.2

Table 2.18: The perplexity of neural LMs. SelfSeg uses subwordMASS strategy.

for the neural model to generate the word, the possible segments themselves are generated by BPE, which are not optimized for sequence generation.

Chapter 3

BERTSeg: BERT-based Unsupervised Subword Segmentation for Low-Resource Neural Machine Translation

Subword segmentation is the task of splitting a word into smaller n-gram character units called subwords [112]. It alleviates the out-of-vocabulary (OOV) problem in neural machine translation (NMT) [126, 4, 135] by enabling an NMT system to have a fixed-size vocabulary while being able to handle all possible words regardless of their frequencies.

Studies in subword segmentation fall into two categories: frequency-based approaches and neural network-based approaches. Frequency-based approaches [116, 62, 61, 106] adopt a greedy algorithm that generates the vocabulary with frequent subword fragments in the corpus during training and merges adjacent high-frequency fragments starting from characters recursively during inference. Among these methods, BPE-dropout [106] and SentencePiece with regularization [61] generate multiple segmentations by random sampling. Frequency-based approaches do not consider semantic information of the subwords, therefore the generated segmentation is not linguistically motivated. For example, the word "fellow-

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

Table 3.1: BERTSeg produces linguistically intuitive subword semgnetations.

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

Table 3.2: BERTSeg-Regularization samples multiple segmentations from one word.

ships" is segmented into "fell/ows/hip/s" by BPE whereas "fellow/ships" is a more linguistically motivated segmentation. Neural approaches such as DPE [41] implicitly considers the contextual semantic information of subwords by maximizing the generation probabilities of the target language sentences conditioned on the source language sentences. However, it trains on parallel sentences, which poses a problem for low-resource languages. DPE is slow because it calculates the probabilities of all possible sentence segmentations, therefore, not practical in high-resource scenarios.

We propose BERTSeg, an unsupervised neural subword segmenter that leverages contextualized word representations from the pre-trained model, characterBERT [29]. It combines the advantages of frequency-based and neural approaches by 1) leveraging word-level monolingual data and 2) capturing semantic information explicitly. The semantic information is provided by characterBERT, which

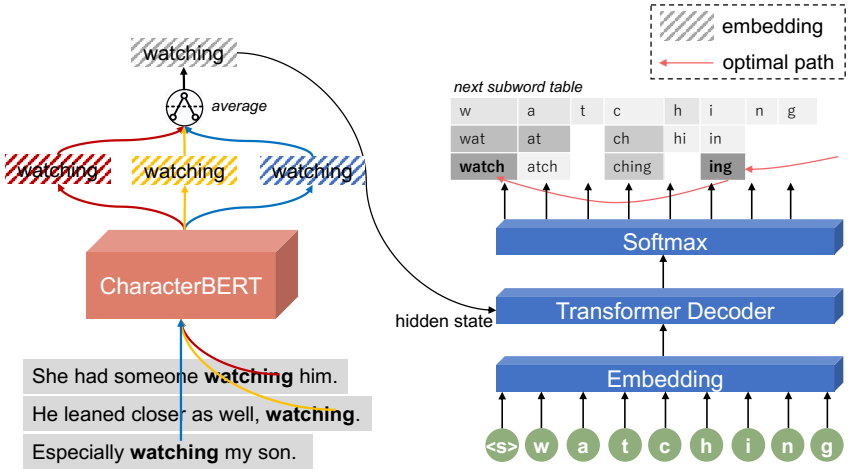


Figure 3.1: **BERTSeg Architecture.** The encoder is a characterBERT that generates average embeddings for one word in different contexts. The transformer decoder takes characters as input and generates probabilities of the next subword. During training, the objective is to maximize the probabilities of all possible segmentations. During inference, the model retraces the optimal segmentation.

has been shown to be helpful for natural language understanding tasks. In our task, this enables the model to generate linguistically intuitive segmentations rather than high-frequency fragments, as shown in Table 3.1.

Furthermore, we propose a subword regularization method named BERTSeg-Regularization which enables the model to produce multiple segmentations based on segmentation probabilities to improve the robustness of NMT, as represented in Table 3.2.

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

3.1 Related Work

Early NMT studies apply word-level vocabulary to represent only frequent words, which causes the out-of-vocabulary (OOV) problem [126]. To address this, character-based [53, 20, 69], hybrid word-character based [72], or UTF-8 based [117] NMT models were proposed. However, the resultant long input/output sequences increase the model and computational complexity.

Subword segmentation methods address the OOV problem by segmenting words into subwords that are in a fixed vocabulary of character n-grams. BPE [116, 31] generates the subword vocabulary by first splitting all the sentences into characters, then iteratively saving the most frequent adjacent pairs into the vocabulary and merging them, until reaching the desired size. Each test sentence is segmented similarly. WordPiece [112] and SentencePiece [62] are another two widely-used subword methods.

Among the subword methods, BPE [116] does not model the input sequence whereas SentencePiece [62] applies a unigram model to output probabilities of each segmentation. Based on sequence modeling via segmentations theory [143], the generation probability of a target sentence can be calculated by the sum of probabilities of all its possible segmentations. DPE [41] models the whole target sentence conditioned on the source sentence. However, we show that modeling words conditioned on their semantic embedding is a more efficient way.

Regularization as data augmentation can boost performance. BPE-dropout [106] randomly drops subword merge operation. SPM-regularization [61] generates multiple segmentations with their probabilities. Leveraging the dynamic programming algorithm, we retrace the global best-n segmentations with the highest probabilities in polynomial time.

3.2 Methodology

We aim to obtain the optimal subword segmentation with the highest generation probability by neural models. First, we introduce the relationship between word probability and subword segmentation probability. Second, we propose a model to maximize the marginal word probability and retrace the optimal seg-

mentation. Additionally, we propose a regularization method to retrace multiple segmentations for one word.

3.2.1 Background: Word Modeling

We define a word as a single distinct meaningful element of writing. Technically, we split words in sentences with tools for different languages as described in Section 3.3. Let $\boldsymbol{x}_{1:T}$ denote a word containing T characters. $\boldsymbol{a}_{1:\tau_a}$ is one segmentation of $\boldsymbol{x}$ that comprises τ_a subwords a_i. $\mathcal{S}(\boldsymbol{x})$ is the set of all possible segmentations of $\boldsymbol{x}$. The generation probability $\boldsymbol{x}$ can be defined as the sum of the probabilities of all segmentations shown in Eq. (2.1).

$$\begin{aligned} p(\boldsymbol{x}_{1:T}) &= \sum_{\boldsymbol{a}_{1:\tau_a} \in \mathcal{S}(\boldsymbol{x})} p(\boldsymbol{a}_{1:\tau_a}) \\ &= \sum_{\boldsymbol{a}_{1:\tau_a} \in \mathcal{S}(\boldsymbol{x})} \prod_{i=1}^{\tau_a} p(a_i | a_1, ..., a_{i-1}) \end{aligned} \tag{3.1}$$

3.2.2 Proposed Method: BERTSeg

As shown in Figure 3.1, the proposed BERTSeg contains a characterBERT encoder [29] and a mixed character-subword transformer decoder [41]. The mixed character-subword transformer takes characters as input and generates sub-words as output. The model represents the history information by prefix characters $x_1, ..., x_j$ instead of previous subwords $a_1, ..., a_{i-1}$, where j is the index of the last character in a_{i-1}.

Let $\boldsymbol{e}_{\boldsymbol{x}}$ denote the average-pooled contextualized word embeddings by characterBERT from all sentences containing word $\boldsymbol{x}$. The generation probability can be calculated by Eq. (3.2).

$$\begin{aligned} &\log p(\boldsymbol{x}_{1:T} | \boldsymbol{e}_{\boldsymbol{x}}) = \\ &\log \sum_{\boldsymbol{a}_{1:\tau_a} \in \mathcal{S}(\boldsymbol{x})} \prod_{i=1}^{\tau_a} p(a_i | \boldsymbol{e}_{\boldsymbol{x}}; x_1, ..., x_j) \end{aligned} \tag{3.2}$$

During training, we calculate the $\log p(\boldsymbol{x}_{1:T} | \boldsymbol{e}_x)$ in polynomial time by dynamic programming (DP) [41] and use $-\log p(\boldsymbol{x}_{1:T} | \boldsymbol{e}_x)$ as the loss. During inference, we

retrace the optimal segmentation $\boldsymbol{a}$ through Eq. (3.3).

$$\boldsymbol{a} = \underset{\boldsymbol{a}_{1:\tau_a} \in \mathcal{S}(\boldsymbol{x})}{\arg\max} \prod_{i=1}^{\tau_a} p(a_i | \boldsymbol{e}_x; x_1, ..., x_j) \tag{3.3}$$

3.2.3 Probability Based Regularization

We propose BERTSeg-Regularization which performs subword regularization based on the probability distribution during inference. For segmentation $\boldsymbol{a}_i$ with $p(\boldsymbol{a}_i)$, the sampling probability $p_{sample}(\boldsymbol{a}_i)$ is shown in Eq. (3.4), where t is a temperature hyperparameter.

$$p_{sample}(\boldsymbol{a}_i) = \frac{e^{\log p(\boldsymbol{a}_i)/t}}{\sum_{\boldsymbol{a}_i \in \mathcal{S}(\boldsymbol{x})} e^{\log p(\boldsymbol{a}_i)/t}} \tag{3.4}$$

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

3.3 Experimental Settings

Datasets Table 3.3 summarizes MT datasets from low- to high-resource. We use the English words of each dataset to train BERTSeg. We applied Juman++ [132] to Japanese sentences, Stanford-segmenter [75] to Chinese sentences, and Moses tokenizer [58] to sentences in other languages. We removed diacritics in Romanian sentences. We set the subword vocabulary size to $8k$ for all segmentation methods and NMT models.

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

Table 3.3: Statistics of the corpora (# sentences).

Segmenter Settings For BERTSeg, we used the characterBERT model [29] trained on English Wikipedia data as encoder, and pre-processed the English data of each dataset to obtain word embeddings. Our transformer decoder was 4-layer with 1 attention head. All hidden sizes in the model were 768. The vocabulary of possible subwords used a BPE vocabulary obtained from the English part of each dataset. To prevent overfitting, we set the gradient clip to 1.0 and trained the model until the loss of $7k$ high-frequency words was stable. BERTSeg-Regularization generated 10 segmentations with the highest probability for each word and t was set to 5. We generated data of each epoch dynamically. Our method was applied to the English sentences, whereas sentences in the other languages used BPE or BPE-dropout.

Baseline methods are BPE, VOLT, DPE, and BPE-dropout. We used the official implementations with default settings of each method for sentences in both source and target languages.

NMT Settings We used the $transformer_{base}$ architecture [135] and the fairseq framework [100]. We trained the model until no BLEU score improvement for 10 epochs on the validation set. During inference, beam size was 12 and length penalty was 1.4. We report sacreBLEU [105] and METEOR [6] on detokenized outputs.

3.4 Results and Analysis

MT Results Tables 3.4, 3.5, 3.6, and 3.7 compare the proposed methods with baseline methods. First, BERTSeg-Regularization achieves the best performance in all directions, significantly boosting BLEU scores up to 8 points and METEOR scores up to 5 points over BPE. Second, regularization is effective: methods with regularization show higher BLEU scores. Among methods w/o regularization, BERTSeg yields the highest BLEU and METEOR scores in most directions. Finally, we found the proposed method especially effective in low-resource scenarios with the help of the pre-trained model trained on large-scale data. As the train set grows, BPE and DPE gradually learn good segmentations, making the gap

	Fil→En	Id→En	Ja→En	Ms→En	Vi→En	Zh→En	Avg
w/o Regularization							
BPE [116]	23.09	25.70	9.42	28.19	19.94	12.21	19.76
VOLT [150]	22.99	25.05	10.56	27.91	21.64	11.31	19.91
DPE [41]	24.04	26.66	9.93	27.89	20.06	10.72	19.88
BERTSeg	24.84$^{*}_{+1.8}$	25.84$_{+0.1}$	10.97$^{*\circ}_{+1.6}$	29.52$^{*\circ}_{+1.3}$	20.86$_{+0.9}$	12.20$^{\circ}_{-0.0}$	20.71$_{+1.0}$
With Regularization							
BPE-dropout [106]	28.18	28.02	12.84	31.59	23.67	13.91	23.04
BERTSeg-Regularization	**31.09**$^{*\circ}_{+8.0}$	**28.86**$^{*\circ}_{+3.2}$	**15.56**$^{*\circ}_{+6.1}$	**32.97**$^{*\circ}_{+4.8}$	**24.58**$^{*\circ}_{+4.6}$	**15.03**$^{*\circ}_{+2.8}$	**24.68**$_{+4.9}$

Table 3.4: **Low-resource Asian languages→English MT BLEU score** results. BERTSeg-Regularization consistently improves over all baselines. Statistical significance $p < 0.001$ is indicated by * against BPE and by $^{\circ}$ against DPE. Subscript values denote the BLEU score differences from BPE.

	Fil→En	Id→En	Ja→En	Ms→En	Vi→En	Zh→En	Avg
w/o Regularization							
BPE [116]	29.05	31.05	20.12	32.74	27.64	22.85	27.24
VOLT [150]	29.16	30.98	21.24	32.50	28.37	22.22	27.41
DPE [41]	29.72	31.79	21.13	32.50	26.94	21.46	27.26
BERTSeg	30.28$_{+1.2}$	31.25$_{+0.2}$	21.04$_{+0.9}$	33.34$_{+0.6}$	27.38$_{-0.3}$	22.57$_{-0.3}$	27.64$_{+0.4}$
With Regularization							
BPE-dropout [106]	31.96	32.99	22.83	34.81	29.05	23.56	29.20
BERTSeg-Regularization	**34.35**$_{+5.3}$	**33.38**$_{+2.3}$	**25.14**$_{+5.0}$	**36.13**$_{+3.4}$	**30.40**$_{+2.8}$	**24.57**$_{+1.7}$	**30.66**$_{+3.4}$

Table 3.5: **Low-resource Asian languages→English MT METEOR score results.** BERTSeg-Regularization consistently improves over all baselines. Subscript values denote the BLEU score differences from BPE.

between BERTSeg smaller.

Training Speeds As presented in Table 3.8, the training speed of BERTSeg is substantially faster than the previous neural method DPE because it trains on word-level data. According to Zipf's law, the number of distinct words in a document increases much slower than the increment of the total number of words. The speed is comparable to non-neural approaches, BPE, and faster than VOLT.

Size of Training Data With the pre-trained encoder, we can train a high-quality segmenter with a tiny train set. We train BERTSeg on words from $500k$ English sentences in the news commentary dataset and apply it to the ALT English

	IWSLT15 Vi→En	WMT16 Ro→En	WMT15 Fi→En
w/o Regularization			
BPE [116]	27.09	32.54	17.45
VOLT [150]	27.16	31.89	17.25
DPE [41]	27.40	29.95	16.14
BERTSeg	$27.80_{+0.7}$	$32.33^{\circ}{}_{-0.2}$	$17.54^{\circ}{}_{+0.1}$
With Regularization			
BPE-dropout [106]	28.76	33.59	**18.50**
BERTSeg-Regularization	$\mathbf{30.09}^{*\circ}{}_{\mathbf{+3.0}}$	$\mathbf{33.82}^{*\circ}{}_{\mathbf{+1.3}}$	$\mathbf{18.46}^{*\circ}{}_{\mathbf{+1.0}}$

Table 3.6: **High-resource MT BLEU score** results. Statistical significance $p < 0.001$ is indicated by * against BPE and by $^{\circ}$ against DPE. Subscript values denote the BLEU score differences from BPE.

words. The averaged BLEU score for MT is 24.45 whereas using only $18k$ ALT English data to train BERTSeg achieved 24.68 points, which are almost the same.

Subword Frequency Distribution Figure 3.2 shows the distribution of subword frequency in the decoded ALT train set of different methods with the same BPE vocabulary. Compared with BPE, BERTSeg generates more high-frequency (> 1000) subwords such as *ed* and *ing*. At the same time, more subwords in the vocabulary are not used during inference (with frequency 0). This phenomenon is also present in the comparison of BERTSeg-Regularization and BPE-dropout. Based on this observation, it is possible to use a smaller vocabulary for BERTSeg. Additionally, we found the total subwords frequency of BERTSeg is higher because sometimes it also segments high-frequency words into subwords such as *years* into *year/s* whereas BPE keeps it as *years*.

Zero-shot Word Segmentations Table 3.9 demonstrates the strong generalization ability on unseen words in the test set. Different from BPE which prefers

	IWSLT15 Vi→En	WMT16 Ro→En	WMT15 Fi→En
w/o Regularization			
BPE [116]	31.16	35.18	27.06
VOLT [150]	30.90	34.90	26.73
DPE [41]	31.07	30.15	26.00
BERTSeg	$31.36_{+0.2}$	$35.16_{-0.0}$	$27.32_{+0.3}$
With Regularization			
BPE-dropout [106]	32.09	35.73	28.39
BERTSeg-Regularization	$\mathbf{32.37}_{+1.2}$	$\mathbf{36.29}_{+1.1}$	$\mathbf{28.61}_{+1.6}$

Table 3.7: **High-resource MT METEOR score results.** Subscript values denote the BLEU score differences from BPE.

high frequency pieces such as *fell* and *hip* in the word *fellowships*, BERTSeg identifies meaningful fragments *fellow* and *ships*.

Evaluation on Segmentations This section provides a direct evaluation of the subword segmentation itself. We use a large-scale morphological database that provides pairs of words and segmentations [7], where several English word examples from this database are shown in Table 3.10. We compared the segmen-

	ALT	WMT16 Ro-En
†BPE [116]	4	13
†VOLT [150]	960	1,747
♢DPE [41]	3,477	68,334
♠**BERTSeg**	58	391

Table 3.8: Training speeds (seconds). †: trained on CPU, ♢: on 8 32GB GPUs, ♠ on 1 12GB GPU.

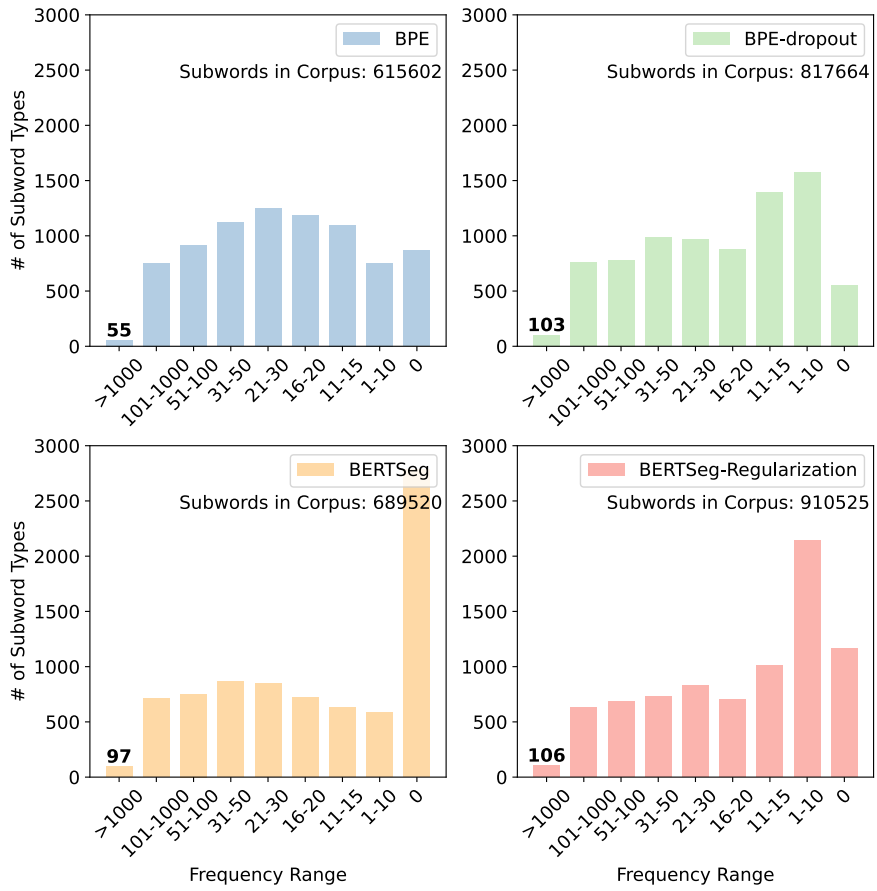


Figure 3.2: Subword frequency distributions of BPE, BPE-dropout, BERTSeg, and BERTSeg-Regularization.

tations by BERTSeg, BPE with those in the MorphyNet database. We found that the exact matching rate of the suffix subword is 50.8% by BERTSeg and 23.3% by BPE, showing that BERTSeg segmentation is more linguistically motivated. The BERTSeg and BPE segmenters are trained on the English data of the WMT'15 Fi–En dataset, and the exact rates of the whole segmentations are low because of the limitation of the subword vocabulary.

We also compared with segmentations in the *Oxford Advanced Learner's Dictionary* by manually collecting 100 subword segmentations (named *word division* in the dictionary). We found that the subword matching rate of BERTSeg with

BERTSeg	**BPE** [116]
fellow/ships	fell/ows/hip/s
re/creation/al	rec/re/ational
dis/claim/er/s	discl/aim/ers
post/season	pos/ts/e/ason
re/fresh/ed	ref/res/hed
worse/n/s	wor/s/ens

Table 3.9: BERTSeg and BPE tested on unseen words.

the dictionary is 81.6% and that of BPE with the dictionary is 53.9%. Some examples are shown in Table 3.11. Only prototypes are contained in the dictionary.

Word	**MorphyNet**	**BERTSeg**	**BPE**
nearer	near/er	near/er	ne/arer
nearest	near/est	near/est	ne/are/st
stonewalling	stonewall/ing	stone/wall/ing	st/one/w/alling
stonewalled	stonewall/ed	stone/wall/ed	st/one/w/alled
foredoors	foredoor/s	fore/doors	fore/d/o/ors
denyings	denying/s	deny/ing/s	deny/ings
missignaled	missignal/ed	mis/signal/ed	miss/ign/al/ed

Table 3.10: Comparing with the annotated MorphyNet segmentations.

3.5 Conclusion and Future Work

We proposed BERTSeg, an unsupervised neural subword segmenter for NMT, together with a regularization algorithm. MT results showed significant improvement over frequency-based and neural network-based methods. The training is efficient even compared with non-neural methods. To address the limitations shown

Word	Dictionary	BERTSeg	BPE
livestock	live/stock	live/stock	li/vest/ock
retouch	re/touch	re/touch	ret/ouch
recall	re/call	re/call	rec/all
massively	mas/sive/ly	massive/ly	mass/ively
abuser	ab/user	abuse/r	ab/us/er
officer	of/fi/cer	office/r	offic/er

Table 3.11: Comparing with segmentations in the Oxford Advanced Learner's Dictionary.

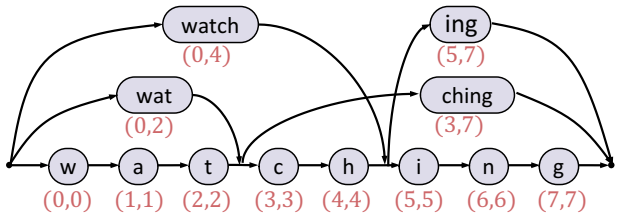
in Appendix A.1, future works include eliminating the dependency on the BPE vocabulary, extending to a multilingual segmenter with mBERT [24] embeddings, and applying it to other generation tasks.

Chapter 4

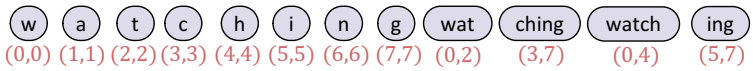
DiverSeg: Leveraging Diverse Segmentations for Low-Resource Neural Machine Translation

A subword segmenter is a crucial preprocessing component in modern neural machine translation (NMT) models [147, 30]. The subword segmenter can address the out-of-vocabulary (OOV) problem [116, 62] as well as enhance the quality of translations compared with non-subword methods [51, 41, 121, 122].

Each type of subword segmenter captures a unique aspect of the input text. The byte-pair encoding (BPE) segmenter [116] based on the byte-pair encoding compression algorithm [31] represents the input text in a highly compressed token sequence [32]. The SentencePiece (SPM) subword segmenter, which is equipped with a unigram language model, captures contextual information to a certain extent [62]. The neural subword segmenter BERTSeg [121] is tuned on pre-trained BERT models and enables the use of semantic meaning and the ability to segment words into linguistically motivated subwords. However, NMT models based on a single segmenter cannot capture various aspects of the input text. This ability is especially important for low-resource NMT because we must make maximum



(a) Subword Lattice.



(b) Flattened Subword Lattice.

Figure 4.1: **Subword Lattice and Flattened Lattice.** The flattened version is suitable for the input of the transformer model where each subword is accompanied by (*start*, *end*) positional information.

usage of limited data.

To overcome this limitation, one approach is to use BPE segmenters of varying vocabulary sizes to generate multiple segmentations and input them into a lattice-aware transformer encoder [148]. This method demonstrates that high diversity in segmentations can improve MT performance. However, this approach uses only one type of segmenter; thus, multi-view information of the input text cannot be captured effectively, and the relations between subwords are incomplete. Another approach uses BPE and SPM segmenters to generate more diverse segmentations for each sentence in the dataset [51], which is compatible with all NMT models because it performs only data-level augmentation. However, it uses the vanilla transformer model [135] which is not aware of any connections between the different segmentations of one word. Moreover, these two approaches lack explicit information sharing among multiple segmentations in the training objective, which results in the embeddings of multiple segmentations of the same word far apart in the embedding space.

In this study, we proposed DiverSeg to address these three problems using three key components: 1) a subword lattice input that contains character, BPE subwords, and BERTSeg subwords with positional information as shown in Fig-

ure 4.1; 2) a subword-relation-aware attention mechanism with fine-grained relation types calculated from the relative position information; 3) a cross-granularity alignment objective that maximizes the cosine similarity among various subword segmentations.

We evaluated the proposed DiverSeg approach on five datasets, including Asian Language Treebank (ALT) [131], IWSLT'15 Vietnamese–English, WMT'16 Romanian–English, WMT'15 Finnish–English, and WMT'14 German–English. Experimental results demonstrate that the proposed approach outperforms baseline methods by approximately two BLEU points. Furthermore, we performed ablation studies to examine the improvement over non-subword methods, contribution of each component, effect of different subword-relation mechanisms in the subword-relation-aware attention, effect of using various similarity measurement metrics in the cross-granularity embedding alignment loss, and selection of the combination of subword segmenters.

The contributions of this study are as follows.

- DiverSeg, a method that leverages diverse segmentations from multiple subword segmenters (Section 4.3), consists of a subword lattice input (Section 4.3.1), a subword-relation-aware attention mechanism (Section 4.3.2), and a cross-granularity embedding alignment objective (Section 4.3.3).
- Experimental results on five datasets that demonstrate the effectiveness of DiverSeg (Section 4.5.1).
- Analysis of how DiverSeg leads to performance improvements (Sections 4.5.3, 4.5.4, 4.5.5, and 4.5.6).

4.1 Related Work

Subword Segmenters. Sennrich et al. [116] proposed BPE for subword segmentation to solve the OOV problem [126]. Word Piece Model (WPM) [112] is a method similar to BPE, but with the distinction that WPM selects the symbol pairs that maximize the likelihood of training data, whereas BPE selects the most frequent pairs. Kudo and Richardson [62] proposed SentencePiece, which

supports a unigram language model and can handle multiple languages, including those without whitespace word boundaries, such as Chinese or Japanese. Byte-level BPE (BBPE) [142] is a variant of BPE that operates on bytes rather than on characters. In VOLT [150], the optimal size of the subword vocabulary is calculated based on the optimal transport algorithm. A survey provided information on non-neural segmentation methods and their limitations [90]. Dynamic programming encoding (DPE) [41] and bilingual subword segmentation [23] have been proposed for NMT. In these methods, both target and source languages are considered. Bilingual subword segmentation is based on a unigram language model, whereas the DPE is based on a transformer model. The BERTSeg [121] and Self-Seg [122] segmenters obtain subword segmentations that maximize the generation likelihood, where they require only word-level monolingual data for training and inference, rendering the model highly efficient. In addition to general-purpose subword segmentation, task-specific subword segmentation methods [44, 45] aim to obtain appropriate tokenizations based on the loss of downstream tasks.

Stochastic subword segmentation (or subword regularization) is an efficient method for improving the robustness of the NMT models. BPE-dropout [106] is a stochastic version of BPE [116] where the merge operations of subwords are randomly dropped during decoding, resulting in various segmentations for each word. SentencePiece regularization [61] is a stochastic version of the SPM [62], which provides multiple segmentations with probabilities for each sentence from a unigram language model. However, these stochastic methods are based on BPE or SPM; therefore, they cannot capture linguistical information, such as BERTSeg [121] or DPE [41] does. The proposed DiverSeg framework combines information from multiple subword segmenters.

Lattice Models. Lattice-aware models are widely used in numerous natural language processing (NLP) tasks such as speech translation [152], machine translation [148], language modeling [10], Chinese Named Entity Recognition (NER) [67], and pre-trained BERT or GPT models [63, 38]. The manner in which models interpret the lattice inputs differ largely according to the target tasks. For the Chinese NER task, using relative position representations [148], which are the difference of positions between any two subwords, achieves comparable per-

formance compared with using pre-defined subword relations [67]. However, for the MT task, using pre-defined subword relations, which are calculated using the start and end positions, can achieve a better performance than using the relative position representations [148]. We improve from the coarse-grained seven types of relations [148] to fine-grained 20 types by 1) separating the many-to-one mappings, where multiple position relationships are mapped into one subword relation type, into one-to-one mappings, and 2) adding new types considering word boundary information.

NMT with Multiple Segmentations. Hierarchical subword features based on BPE with various vocabulary sizes have been incorporated into RNN-based models [93]. Kambhatla et al. [51] proposed combining multiple subword segmentations in the training set by copying sentences segmented from various segmenters. Xiao et al. [148] proposed using BPE with different vocabulary sizes to generate multiple segmentations and Takase et al. [128] proposed leveraging multiple subword segmentations for subword-regularized models as a model ensemble approach. In contrast to previous models, the proposed subword-relation-aware attention mechanism leverages multiple relation matrices to improve its ability of the model to capture complex subword-level relationships. The proposed cross-granularity alignment explicitly enhances the information sharing between different segmentations.

4.2 Preliminary

This section provides an overview of the standard transformer-encoder architecture [135] that is the basis for our model. The subsequent section details the components in DiverSeg.

Let $x = (x_1, x_2, ..., x_n)$ be the input sequence of subword tokens, where n is the sequence length. The embedding layer maps each token x_i to a vector $\mathbf{e}_i \in \mathbb{R}^d$, where d is the embedding dimension. The positional encoding (PE) layer [135]

adds a vector $\mathbf{p}_i \in \mathbb{R}^d$ to each embedding $\mathbf{e}_i$, where $\mathbf{p}_i$ is computed as follows:

$$p_{i,2j} = \sin\left(\frac{i}{10000^{2j/d}}\right) \tag{4.1a}$$

$$p_{i,2j+1} = \cos\left(\frac{i}{10000^{2j/d}}\right) \tag{4.1b}$$

where $j = 0, 1, ..., \frac{d}{2} - 1$ and the input of the transformer encoder is $\mathbf{Z}_0 = (\mathbf{z}_{0,1}, \mathbf{z}_{0,2}, ..., \mathbf{z}_{0,n})$, where $\mathbf{z}_{0,i} = \mathbf{e}_i + \mathbf{p}_i$.

The transformer encoder consists of L encoder layers with identical structures and non-sharing parameters, where each encoder layer consists of two sub-layers: a multi-head self-attention layer and a feed-forward layer. The multi-head self-attention layer computes the weighted sum of the input vectors, where the weights are determined by the similarity between the input vectors. The similarity is measured by the dot product of the input vectors and scaled by the square root of the dimension. The input vectors are also projected onto various subspaces using linear transformations to allow the model to attend to different aspects of the input. Each head in one layer performs the following computations:

$$\mathbf{Q} = \mathbf{Z}_{l-1}\mathbf{W}^{\mathrm{Q}} \tag{4.2}$$

$$\mathbf{K} = \mathbf{Z}_{l-1}\mathbf{W}^{\mathrm{K}} \tag{4.3}$$

$$\mathbf{V} = \mathbf{Z}_{l-1}\mathbf{W}^{\mathrm{V}} \tag{4.4}$$

$$\mathbf{A}_{\text{weight}} = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d/h}}) \tag{4.5}$$

$$\mathbf{A}_{\text{value}} = \mathbf{A}_{\text{weight}}\mathbf{V} \tag{4.6}$$

where h is the number of heads, d/h is the dimension of each head, $\mathbf{W}^{\mathrm{Q}}, \mathbf{W}^{\mathrm{K}}$, and $\mathbf{W}^{\mathrm{V}} \in \mathbb{R}^{d \times d/h}$ are the projection matrices to generate the query, key, and value matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d/h}$.

The output of the l_{th} encoder layer $\mathbf{Z}_l \in \mathbb{R}^{n \times d}$ is then obtained by concatenating the $\mathbf{A}_{\text{value}}$ of all heads and multiplying the output projection matrix $\mathbf{W}^{\mathrm{O}} \in \mathbb{R}^{d \times d}$ and fed into a feed-forward layer. The output of the transformer encoder is $\mathbf{Z}_L$, where L denotes the number of encoder layers. Each sub-layer is with residual connection [40] and layer normalization [3].

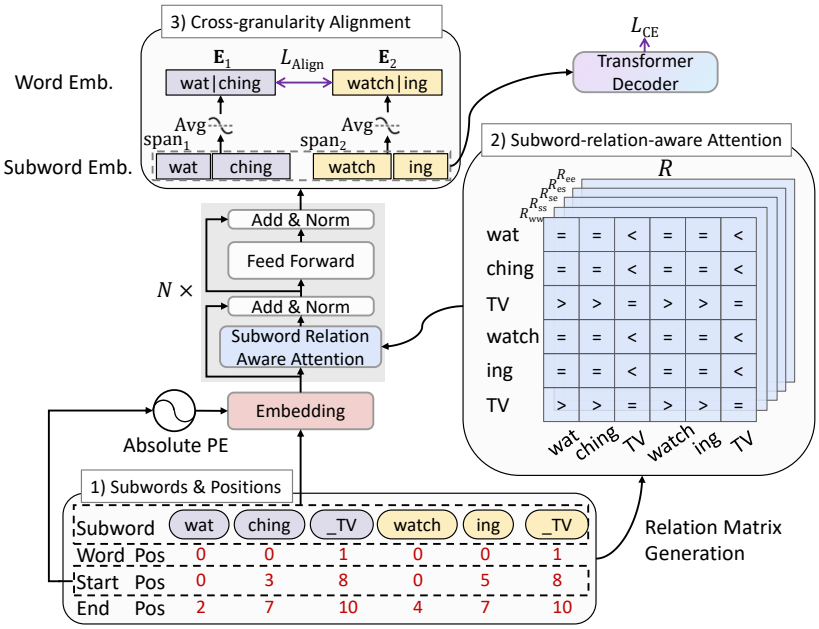


Figure 4.2: **Model Overview**. 1) Segmentations from multiple segmenters are concatenated and accompanied by positions. 2) Relation matrices are generated from positions and provided to the subword-relation-aware attention mechanism. 3) Different segmentations of each word are aligned. Only two segmenters are presented for simplicity.

4.3 Method

DiverSeg contains three parts that differ from the conventional transformer model [135]: 1) the subword lattice input with the subword-relation matrix represents diverse segmentations (Section 4.3.1), 2) the subword-relation-aware attention mechanism captures the relations between any two subwords (Section 4.3.2), and 3) the cross-granularity alignment objective explicitly aligns multiple segmentations (Section 4.3.3).

4.3.1 Multiple Segmentation Representations

We use diverse subword segmentation methods, including character-level segmentation, BPE [116] and BERTSeg [121], to generate multiple segmentations in Di-

Type	$p_i^{\mathrm{w}}, p_j^{\mathrm{w}}$	$p_i^{\mathrm{s}}, p_j^{\mathrm{s}}$	$p_i^{\mathrm{s}}, p_j^{\mathrm{e}}$	$p_i^{\mathrm{e}}, p_j^{\mathrm{s}}$	$p_i^{\mathrm{e}}, p_j^{\mathrm{e}}$	Type	$p_i^{\mathrm{w}}, p_j^{\mathrm{w}}$	$p_i^{\mathrm{s}}, p_j^{\mathrm{s}}$	$p_i^{\mathrm{s}}, p_j^{\mathrm{e}}$	$p_i^{\mathrm{e}}, p_j^{\mathrm{s}}$	$p_i^{\mathrm{e}}, p_j^{\mathrm{e}}$
1	$<$	$<$	$<$	$<$	$<$	**11**	$=$	$=$	$<$	$>$	$>$
2	$=$	$<$	$<$	$<$	$<$	**12**	$=$	$=$	$=$	$=$	$=$
3	$=$	$<$	$<$	$=$	$<$	**13**	$=$	$=$	$=$	$>$	$>$
4	$=$	$<$	$<$	$=$	$=$	**14**	$=$	$>$	$<$	$>$	$<$
5	$=$	$<$	$<$	$>$	$<$	**15**	$=$	$>$	$<$	$>$	$=$
6	$=$	$<$	$<$	$>$	$=$	**16**	$=$	$>$	$<$	$>$	$>$
7	$=$	$<$	$<$	$>$	$>$	**17**	$=$	$>$	$=$	$>$	$=$
8	$=$	$=$	$<$	$=$	$<$	**18**	$=$	$>$	$=$	$>$	$>$
9	$=$	$=$	$<$	$>$	$<$	**19**	$=$	$>$	$>$	$>$	$>$
10	$=$	$=$	$<$	$>$	$=$	**20**	$>$	$>$	$>$	$>$	$>$

Table 4.1: **Subword Relations.** Relation of two subwords is calculated by the orders of word index p^{w}, start index p^{s}, and end index p^{e} of two subwords x_i and x_j.

verSeg and represent them by pre-defined 20 fine-grained subword relation types, where previous work [148] only used BPE segmenter and used seven coarse-grained subword relation types.

In the implementation of DiverSeg, we use *absolute positional encoding* [135] to encode the positional information of each subword and a variant of *relative positional encoding* [118] to encode the information between any two subwords, which is saved in a subword-relation matrix.

Absolute Positional Encoding. A subword lattice structure (Figure 4.1) is used to represent a word with multiple segmentations. Each edge represents a subword in the lattice (formally, a directed acyclic graph). We convert the lattice structure into a sequence structure to fit the input format of the transformer encoder. We flatten the lattice by concatenating multiple segmentations and assign each subword an absolute position denoted by its start position, which is calculated using Eq. (4.1).

Relative Positional Encoding. In addition to the start position, each subword carries two other types of positional information, the end position and word position, with all indices starting from 0, as shown in Figure 4.2.

- word position p_i^{w}, the index of the word that the subword belongs to.

- start position p_i^{s}, the index of the character that the subword starts with.
- end position p_i^{e}, the index of the character that the subword ends with.

Subword-relation Matrix represents the relative relation between two subwords, which is generated from five simple types of relative positional relations of any two subword pairs.

- word–word relation $r_{\mathrm{ww}}^{i,j} = \mathbf{ORF}(p_i^{\mathrm{w}}, p_j^{\mathrm{w}})$
- start–start relation $r_{\mathrm{ss}}^{i,j} = \mathbf{ORF}(p_i^{\mathrm{s}}, p_j^{\mathrm{s}})$
- start–end relation $r_{\mathrm{se}}^{i,j} = \mathbf{ORF}(p_i^{\mathrm{s}}, p_j^{\mathrm{e}})$
- end–start relation $r_{\mathrm{es}}^{i,j} = \mathbf{ORF}(p_i^{\mathrm{e}}, p_j^{\mathrm{s}})$
- end–end relation $r_{\mathrm{ee}}^{i,j} = \mathbf{ORF}(p_i^{\mathrm{e}}, p_j^{\mathrm{e}})$

where the **o**rder relation function ($\mathbf{ORF}(\cdot,\cdot)$) returns the order relation of the first input to the second input $\in \{<, =, >\}$. We generate the subword relation of two subwords x_i and x_j through Table 4.1, which is a look-up table that maps five positional relations to one relation type. We then obtain subword-relation matrix $\mathbf{R} \in \{1, ..., 20\}^{n \times n}$. Comparing with previous work [148], we involve word boundary information in **Type** 1 and **Type** 20. We also separate the coarse-grained many-to-one conditions into fine-grained one-to-one conditions. For instance, relation **inc** ($p_i^{\mathrm{s}} \leq p_j^{\mathrm{s}} < p_j^{\mathrm{e}} \leq p_i^{\mathrm{e}}$) in the previous work [148] is separated into **Type** 6, **Type** 7, **Type** 10, **Type** 11 in our definition.

4.3.2 Subword-relation-aware Attention

DiverSeg uses the same relation-aware self-attention architecture as in Shaw et al. [118]. However, DiverSeg uses fine-grained subword relations as shown in Table 4.1 whereas clipped relative position is used in Shaw et al. [118] and coarse-grained subword relation is used in Xiao et al. [148].

The proposed DiverSeg model leverages the subword-relation matrix $\mathbf{R}$ through a subword-relation-aware attention mechanism, as illustrated in Figure 4.3, where

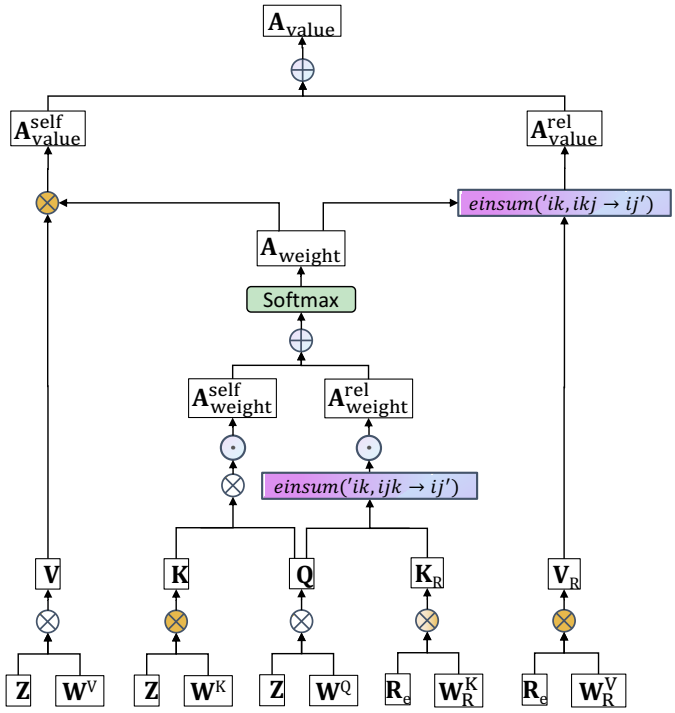


Figure 4.3: **Computational Graph.** This shows the subword-relation-aware attention in each head where all calculation modules are colored. The graph is *symmetric*, where the self-attention mechanism is on the left and the subword-relation-aware attention mechanism is on the right. Variables in the middle are generated jointly by two attention mechanisms. The embedding and dropout layers are omitted for simplicity.

$\mathbf{R}$ provides an additional attention weight matrix $\mathbf{A}^{\text{rel}}_{\text{weight}}$ and value matrix $\mathbf{A}^{\text{rel}}_{\text{value}}$. The calculation in each head per layer is as follows:

Dimensional Transformation. To obtain input vectors with the required dimension in the attention head, we transform input $\mathbf{Z}_{l-1}$ to $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ and $\mathbf{R}$ to $\mathbf{K}_{\text{R}}$, $\mathbf{V}_{\text{R}}$. Eq. (4.10) maps the matrix of discrete types $\mathbf{R}$ into a matrix in the

continuous vecter space $\mathbf{R}_{\mathrm{e}} \in \mathbb{R}^{n \times n \times d}$.

$$\mathbf{Q} = \mathbf{Z}_{l-1}\mathbf{W}^{\mathrm{Q}} \tag{4.7}$$

$$\mathbf{K} = \mathbf{Z}_{l-1}\mathbf{W}^{\mathrm{K}} \tag{4.8}$$

$$\mathbf{V} = \mathbf{Z}_{l-1}\mathbf{W}^{\mathrm{V}} \tag{4.9}$$

$$\mathbf{R}_{\mathrm{e}} = \mathrm{Emb}(\mathbf{R}) \tag{4.10}$$

$$\mathbf{K}_{\mathrm{R}} = \mathbf{R}_{\mathrm{e}}\,\mathbf{W}_{\mathrm{R}}^{\mathrm{K}} \tag{4.11}$$

$$\mathbf{V}_{\mathrm{R}} = \mathbf{R}_{\mathrm{e}}\,\mathbf{W}_{\mathrm{R}}^{\mathrm{V}} \tag{4.12}$$

Attention Weight Matrix The attention weight matrix $\mathbf{A}_{\mathrm{weight}} \in \mathbb{R}^{n \times n}$ is from the self-attention weight and the subword-relation-aware attention weight, as expressed in Eq. (4.15). Here, $\mathbf{A}_{\mathrm{weight}}^{\mathrm{self}}$ is the conventional self-attention weight matrix. Eq. (4.14) calculates the subword-relation-aware attention weight matrix given the query $\mathbf{Q}$ and subword-relation-aware key $\mathbf{K}_{\mathrm{R}}$. The calculation process differs from those in Eq. (4.13) because the shapes of $\mathbf{K} \in \mathbb{R}^{n \times d/h}$ and $\mathbf{K}_{\mathrm{R}} \in \mathbb{R}^{n \times n \times d/h}$ differs. We used the Einstein summation (einsum) operation to sum over along the desired indices, where $\mathbf{A}_{\mathrm{weight}_{(i,j)}}^{\mathrm{rel}} = \sum_k \mathbf{Q}_{(i,k)}\mathbf{K}_{\mathrm{R}_{(i,j,k)}}$.

$$\mathbf{A}_{\mathrm{weight}}^{\mathrm{self}} = \frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d/h}} \tag{4.13}$$

$$\mathbf{A}_{\mathrm{weight}}^{\mathrm{rel}} = \frac{\mathrm{einsum}('ik, ijk \rightarrow ij', \mathbf{Q}, \mathbf{K}_{\mathrm{R}})}{\sqrt{d/h}} \tag{4.14}$$

$$\mathbf{A}_{\mathrm{weight}} = \mathrm{softmax}(\mathbf{A}_{\mathrm{weight}}^{\mathrm{self}} + \mathbf{A}_{\mathrm{weight}}^{\mathrm{rel}}) \tag{4.15}$$

Attention Value Matrix. Eq. (4.16) calculates the attention value $\mathbf{A}_{\mathrm{value}}$ from the self-attention value $\mathbf{A}_{\mathrm{weight}}\mathbf{V}$ and the subword-relation-aware value $\mathbf{A}_{\mathrm{value}}^{\mathrm{rel}} = \mathrm{einsum}('ik, ikj \rightarrow ij', \mathbf{A}_{\mathrm{weight}}, \mathbf{V}_{\mathrm{R}})$, where each element in it is calculated by $\mathbf{A}_{\mathrm{value}_{(i,j)}}^{\mathrm{rel}} = \sum_k \mathbf{A}_{\mathrm{weight}_{(i,k)}}\mathbf{V}_{\mathrm{R}_{(i,k,j)}}$. The order of indices differs from that in Eq. (4.14).

$$\mathbf{A}_{\mathrm{value}} = \mathbf{A}_{\mathrm{weight}}\mathbf{V} + \mathrm{einsum}('ik, ikj \rightarrow ij', \mathbf{A}_{\mathrm{weight}}, \mathbf{V}_{\mathrm{R}}) \tag{4.16}$$

All heads in one layer share $\mathbf{W}_{\mathrm{R}}^{\mathrm{K}} \in \mathbb{R}^{d \times d/h}$ and $\mathbf{W}_{\mathrm{R}}^{\mathrm{V}} \in \mathbb{R}^{d \times d/h}$. Therefore, $\mathbf{K}_{\mathrm{R}} \in \mathbb{R}^{n \times n \times d/n}$, $\mathbf{V}_{\mathrm{R}} \in \mathbb{R}^{n \times n \times d/n}$, and the subword-relation-aware relative attention weight matrix $\mathbf{RelAttn}_{\mathrm{w}}$ are shared across all heads. The remaining parts

such as the concatenation of multiple heads, feed-forward, residual connection, and layer normalization keep the same as the traditional transformer encoder.

4.3.3 Cross-granularity Alignment Objective

To address the lack of explicit information sharing among multiple segmentations in previous studies [148, 51], we propose a cross-granularity embedding alignment loss $\mathcal{L}_{\text{Align}}$ that enables information sharing among various types of segmentations of one word.

As presented in *1) Subword & Positions* in Figure 4.2, the input subword sequences x are generated by concatenating segmentations from s segmenters. Therefore, s spans of consecutive subwords represent one segmentation of word w. As shown in the *3) Cross-granularity Alignment* in Figure 4.2, the representation of one segmentation $\mathbf{E}$ is calculated by averaging the embeddings of the subwords in that span, where each subword embedding is from the last encoder layer.

We define $\mathcal{L}^{\text{w}}_{\text{Align}}$ for word w by the average of cosine similarities ($\cos\langle\cdot,\cdot\rangle$) between each two segmentation representations as follows:

$$\mathcal{L}^{\text{w}}_{\text{Align}} = -\frac{1}{s^2}\sum_{i=1}^{\text{s}}\sum_{j=1}^{\text{s}}\cos\langle\mathbf{E}_i,\mathbf{E}_j\rangle \tag{4.17}$$

and $\mathcal{L}_{\text{Align}}$ for one sentence containing n words by summing the losses of each word w_k:

$$\mathcal{L}_{\text{Align}} = \sum_{k=1}^{n}\mathcal{L}^{\text{w}_k}_{\text{Align}} \tag{4.18}$$

The overall $\mathcal{L}$ can be expressed as a summation for the alignment loss and the cross-entropy loss of the translation task $\mathcal{L}_{\text{CE}}$ with a weight ratio α with 1 as the default setting:

$$\mathcal{L} = \mathcal{L}_{\text{Align}} + \alpha\mathcal{L}_{\text{CE}} \tag{4.19}$$

4.4 Experiment Settings

4.4.1 Datasets

In the NMT experiments, we used the ALT [131], IWSLT'15 English–Vietnamese (En–Vi), WMT'16 English–Romanian (En–Ro), WMT'15 English–Finnish (En–Fi), and WMT'14 English–German (En–De) datasets. The details of the datasets are presented in Table 4.2, which contains the sizes of the datasets and type-token ratios, calculated by $\frac{\#\text{word types}}{\#\text{words}}$.

ALT is a multi-way parallel dataset that contains data in English and multiple Asian languages. We experimented with Filipino (Fil), Indonesian (Id), Japanese (Ja), Malay (Ms), Vietnamese (Vi), and simplified Chinese (Zh). We used the ALT-standard-split tool[1] to split the dataset into training, validation, and test sets. For the WMT'16 English–Romanian dataset, we performed normalization to the Romanian-side data and removed diacritics following previous work [115]. We prepared the WMT'14 English–German dataset using a data clean and normalization tool in Fairseq.[2]

We performed tokenization on all data. We applied Juman++ [132] to Japanese data, the Stanford-tokenizer [75] to Chinese data, and the Moses tokenizer [58] to data in other languages.

4.4.2 DiverSeg Settings

DiverSeg leverages multiple segmenters on the source side. Because our focus was on the encoder side, we applied the widely used BPE segmenter to the data in the target language, including baselines. For *English→the other language* translation direction, DiverSeg uses the character-level segmenter, BPE, and BERT-Seg, where the latter two are trained on English data in the training set of each dataset. For *the other language→English* translation direction, DiverSeg uses the character-level segmenter, BPE, and SentencePiece with a unigram language model, where the latter two are trained on data in the other language in the

[1] Provided here: `www2.nict.go.jp/astrec-att/member/mutiyama/ALT`

[2] `https://github.com/facebookresearch/fairseq/blob/main/examples/translation/prepare-wmt14en2de.sh`

Dataset	Train	Valid	Test	En TTR	XX TTR
ALT En–Asian Langs	18,088	1,000	1,018	6.35%	3.90%
IWSLT'15 En–Vi	133,166	1,553	1,268	2.00%	0.77%
WMT'16 En–Ro	612,422	1,999	1,999	0.56%	0.79%
WMT'15 En–Fi	1,754,754	1,500	1,370	0.27%	2.23%
WMT'14 En–De	4,532,411	45,781	3,003	0.72%	0.56%

Table 4.2: **Statistics of the MT datasets.** Numbers indicate parallel sentences in each set of each dataset. En TTR and XX TTR indicate the type-token ratios for data in English data and data in the other language (averaged in ALT) in each dataset.

training set of each dataset. We used SentencePiece rather than BERTSeg in *the other language→English* direction because BERTSeg, which is based on the English characterBERT currently only supports English.

The BPE, BERTSeg, and SentencePiece segmenters share the same vocabulary in the same direction in each dataset. We limited the sizes of both the source and target vocabularies for all models to 8,000, which are near the optimal sizes provided by the VOLT algorithm [150] shown in Appendix B.1.

4.4.3 NMT Settings

We implemented DiverSeg using the fairseq framework.[3] The model architectures for the ALT dataset were determined through parameter searching on the number of encoder layers, decoder layers, and attention heads. The optimal settings were six encoder layers, six decoder layers, and one attention head. For other datasets, we used a standard `Transformer Base` architecture with six encoder layers, six decoder layers, and eight attention heads. We used the default pre-normalization in fairseq instead of post-normalization in the original transformer paper [135] because pre-normalization exhibits faster training [149, 86]. The dropout and

[3] `github.com/facebookresearch/fairseq`

attention dropout rates were set to 0.1.

During training, each batch in one GPU contains $3,072$ tokens in the source language. We used eight GPUs in all experiments, resulting in approximately 25k source tokens per training batch. We used the Adam optimizer [54], with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. We used warmup and decay strategy for the learning rate following Vaswani et al. [135], with warmup steps of $4,000$, initial learning rate to $1.7 * 10^{-7}$ and learning rate to 0.0005. We used label smoothing for the cross-entropy loss with $\epsilon_{ls} = 0.1$ [127]. We calculated the loss in the validation step after each epoch and applied early stopping when no improvement was observed after 10 epochs.

During inference, we used the checkpoint with the best loss on the validation set. We used beam search with a beam size of 12 and length penalty $\alpha = 1.4$. We evaluated the translation quality through BLEU [101], BLEURT [113], METEOR [6] scores and performed significant test [56] against all baselines.

4.4.4 Baseline Settings

We compared DiverSeg o with three types of baseline methods: 1) models using data segmented by one type of subword segmenter, 2) models using data segmented by multiple types of segmenters, and 3) models using stochastic subword segmenter.

Single-segmenter Methods We compared the proposed model to the BPE [116], SentencePiece [62], VOLT [150], DPE [41], and BERTSeg [121] methods. BERTSeg and DPE were applied only to English data.

Multiple-segmenter Methods We compared with 1) using multiple segmentations for each source sentence in the dataset [51], where we used character-level segmentation, BPE, and BERTSeg in *English→the other language* direction and use character-level segmentation, BPE and SentencePiece in *the other language→English* direction, keeping the same with DiverSeg. During inference, we tested the input sentences segmented by each segmenter and reported the average scores. Detailed results are reported in Table B.2 in Appendix B.2. 2) using the BPE segmenter with multiple vocabulary sizes, where we used character-level

segmentation, BPE with vocabulary sizes of 8k and 16k [148].[4]

Stochastic Subword Segmentation Method We present results of the BPE-dropout method [106]. We applied the algorithm with a default dropout rate of 0.1 to the training set of each dataset except for the high-resource WMT'14 En–De dataset, where the dropout was set to a lower value 0.05 that achieved a better performance on the validation set.

We compared DiverSeg with each subword segmenter it used and with multiple-segmenter methods that used the same subword segmenters. Furthermore, we verified using BPE-dropout as one of the segmenters in DiverSeg. For this setting, we applied the combination of character-level segmentation and BPE-dropout. An analysis of using different segmenter combinations is presented in Section 4.5.6.

4.5 Experimental Results

Section 4.5.1 presents results of the translation quality evaluation, Section 4.5.2 compares with non-subword methods, Section 4.5.3 conducts an ablation study on the three proposed modules, Section 4.5.4 explores various representations of subword relations, Section 4.5.5 examines different alignment loss choices, and Section 4.5.6 analyzes results from various segmenter combinations.

4.5.1 Main Results

We compared DiverSeg with the single- and multiple-segmenter baselines. The BLEU scores of DiverSeg and the baselines are reported in Tables 4.3 and 4.4. DiverSeg outperformed baseline models in most translation directions across all datasets. Specifically, when using the combination of character-level data, BPE and BERTSeg, DiverSeg, DiverSeg achieves 3.59 and 4.52 BLEU score compared with the best multiple-segmenter method [148] and the BPE baseline, respectively, on average across the ALT dataset in the English→Asian languages translation directions. When using the combination of character-level data, BPE, and Sen-

[4]Because the code is not public [148], we implemented the method by ourselves, where we use richer subword relations from Table 4.1 (except types 1 and 20, which require word boundary information) than in their paper.

	En–Fil		En–Id		En–Ja		En–Ms		En–Vi		En–Zh		Avg.	
	→	←	→	←	→	←	→	←	→	←	→	←	→	←
Single-Segmenter														
BPE	18.85	16.42	22.34	19.02	10.40	5.89	26.49	21.37	20.95	15.18	7.80	6.97	17.81	14.14
SentencePiece	19.64	16.64	21.92	18.74	**10.97**	5.74	25.90	21.01	19.69	13.93	7.13	7.36	17.54	13.90
VOLT	18.37	16.83	22.34	19.02	10.40	5.89	26.90	21.12	21.15	14.94	8.04	7.00	17.87	14.13
DPE	19.23	15.77	22.81	19.71	10.45	5.97	27.46	22.70	20.02	15.16	8.36	7.44	18.06	14.46
BERTSeg	20.08	17.75	21.87	20.04	10.93	6.56	25.46	21.92	20.95	15.51	8.07	7.91	17.89	14.95
Multiple-Segmenter														
Kambhatla et al. [51]	13.54	11.06	14.99	13.16	7.30	5.58	18.79	15.24	14.29	10.79	3.84	6.96	12.13	10.47
Xiao et al. [148]	20.86	**18.79**	23.70	19.81	10.46	5.87	27.10	24.05	22.32	**16.91**	8.01	7.64	18.74	15.51
DiverSeg	♣**22.89**	18.28	♣**25.22**	♣**20.80**	10.72	♣**7.73**	♣**30.51**	♣**26.28**	♣**24.02**	16.49	**8.60**	♣**8.52**	♣**20.33**	♣**16.35**
Stochastic Method														
BPE-dropout	29.14	20.40	31.47	28.31	13.48	11.12	35.93	32.03	27.86	21.96	9.98	12.42	24.64	21.04
DiverSeg w/ BPE-dropout	♣**30.61**	♣**30.29**	♣**34.27**	♣**30.45**	♣**17.38**	♣**13.17**	♣**39.52**	**32.28**	♣**30.29**	♣**24.80**	♣**12.88**	♣**13.90**	♣**27.49**	♣**24.15**

Table 4.3: **BLEU results on the ALT dataset.** The best result in each group is highlighted in **bold**. Statistical significance is assessed for each individual direction, with a significance level of $p < 0.01$ denoted by ♣. We compared DiverSeg to the best-performing method in each direction and contrast DiverSeg w/ BPE-dropout against BPE-dropout. For the *Avg.* score's statistical significance, we concatenated all translated files and compared them against the combined files of the *one best* baseline method that had the highest *Avg.* score.

	IWSLT'15 En–Vi		WMT'16 En–Ro		WMT'15 En–Fi		WMT'14 En–De	
	→	←	→	←	→	←	→	←
Single-Segmenter								
BPE	26.23	22.36	32.48	31.93	14.94	17.10	25.13	30.27
SentencePiece	26.60	22.68	32.10	32.24	14.97	17.45	25.30	30.00
VOLT	25.87	22.28	31.82	31.54	14.27	17.06	25.35	**30.56**
DPE	26.45	23.78	32.07	31.89	14.67	17.44	25.30	30.55
BERTSeg	26.83	23.10	32.36	32.24	**15.07**	17.83	24.74	29.71
Multiple-Segmenter								
Kambhatla et al. [51]	27.14	23.89	32.52	32.34	15.04	17.79	24.96	30.33
Xiao et al. [148]	28.67	24.91	32.08	31.29	13.73	16.44	25.26	30.07
DiverSeg	♠**29.19**	♠**25.78**	**32.68**	**32.47**	14.89	**17.85**	**25.58**	30.36
Stochastic Method								
BPE-dropout	30.29	**28.06**	**34.91**	**34.76**	**16.32**	**19.05**	24.93	30.39
DiverSeg w/ BPE-dropout	♠**30.84**	28.03	34.62	34.07	15.97	18.94	-	-

Table 4.4: **BLEU results on the IWSLT'15 and WMT datasets.** The highest score in each group is highlighted in **bold**. Statistical significance is assessed for each individual direction, with a significance level of $p < 0.05$ denoted by ♠. We compare DiverSeg to the best-performing method in each direction and also contrast DiverSeg w/ BPE-dropout with BPE-dropout.

	ALT En–Langs		IWSLT'15 En–Vi		WMT'16 En–Ro		WMT'15 En–Fi		WMT'14 En–De	
	→	←	→	←	→	←	→	←	→	←
Single-Segmenter										
BPE	33.91	38.37	51.31	54.09	66.23	61.91	61.74	56.05	60.30	64.56
SentencePiece	33.29	37.80	51.67	53.98	66.09	62.29	62.35	56.58	60.17	64.41
VOLT	33.89	38.20	50.93	53.68	65.27	61.11	60.30	55.43	60.31	**65.08**
DPE	34.29	39.02	51.23	54.73	65.92	62.37	62.57	56.36	60.19	64.68
BERTSeg	34.32	39.61	51.71	55.03	65.78	62.13	62.53	55.98	59.38	64.33
Multiple-Segmenter										
Kambhatla et al. [51]	27.82	32.21	51.92	53.94	66.85	62.67	**63.54**	**57.81**	59.79	64.64
Xiao et al. [148]	34.87	39.10	55.28	56.06	66.43	61.36	60.57	54.80	60.35	64.81
DiverSeg	**37.44**	**40.57**	**56.20**	**56.24**	**66.94**	**62.90**	62.85	56.62	**61.09**	64.48
Stochastic Method										
BPE-dropout	43.29	46.64	57.11	**59.52**	**70.02**	**65.24**	**65.15**	**58.96**	59.42	64.82
DiverSeg w/ BPE-dropout	**46.68**	**49.97**	**58.57**	59.18	69.26	64.98	64.36	57.97	-	-

Table 4.5: **BLEURT Results.** The best result in each group is in highlighted in **bold**.

	ALT En–Langs		IWSLT'15 En–Vi		WMT'16 En–Ro		WMT'15 En–Fi		WMT'14 En–De	
	→	←	→	←	→	←	→	←	→	←
Single-Segmenter										
BPE	26.25	24.45	32.09	29.49	**31.17**	35.11	19.79	27.08	28.33	33.81
SentencePiece	25.85	24.28	32.35	29.26	31.10	35.20	**19.99**	27.42	28.34	33.57
VOLT	26.15	24.42	32.83	28.93	30.75	34.94	19.27	26.78	**28.46**	33.87
DPE	26.45	24.65	32.46	29.65	31.02	35.33	19.69	27.62	28.28	33.76
BERTSeg	26.21	24.51	32.70	29.83	30.90	35.33	19.86	27.31	28.05	33.54
Multiple-Segmenter										
Kambhatla et al. [51]	21.26	20.18	32.56	29.19	31.15	35.36	19.91	**27.83**	28.15	33.89
Xiao et al. [148]	25.93	24.62	33.44	**30.51**	30.87	34.67	19.05	26.29	28.14	**34.30**
DiverSeg	**26.51**	**25.34**	**33.78**	30.41	31.09	**35.50**	19.74	27.50	28.28	34.05
Stochastic Method										
BPE-dropout	29.16	28.83	34.14	**32.08**	**32.39**	**36.74**	**20.91**	**28.92**	26.91	33.99
DiverSeg w/ BPE-dropout	**30.98**	**30.86**	**34.36**	32.01	32.15	36.35	20.71	28.29	-	-

Table 4.6: **METEOR Results.** The best result in each group is in highlighted in **bold**.

tencePiece, the improvement brought by the DiverSeg becomes smaller than when involving BERTSeg, while still better than all baselines on the ALT dataset in the Asian languages→English translation direction. For the MultiSub method [51], we report the average scores using input sentences segmented by each segmenter and provide a detailed analysis in Table B.2 in Appendix B.2. However, the improvement of DiverSeg over other methods on larger datasets, such as WMT'15 En–Fi and WMT'14 En-De, was not as significant as that on small datasets. The gap between all the segmentation methods decreased for large datasets. We hypothesized that this phenomenon could occur because multiple segmentation methods increase the number of times each subword is trained as a data augmentation method, and on large datasets, each subword is trained sufficiently. Additionally, we observed that for source languages with higher type-token ratios (as shown in Table 4.2), DiverSeg yielded more significant BLEU score improvements. For example, for the IWSLT'15 En–Vi dataset, where the type-token ratio of the English data is higher, we observe a more considerable improvement in the En→Vi direction. For the WMT'15 En–Fi dataset, we can observe a more considerable improvement in the Fi→En direction because Finnish is a morphologically rich language, and the type-token ratio is higher than that of English.

We also present the results of using the stochastic subword segmentation method BPE-dropout [106], as well as the stochastic version of DiverSeg, where we incorporated both character-level segmentation and BPE-dropout segmentation. In low-resource scenarios, BPE-dropout showed impressive performance, and DiverSeg with BPE-dropout showed even better results, achieving +3 BLEU scores. Conversely, in high-resource scenarios, the performances of the two stochastic methods were comparable. We abstained from experiments on the WMT14 dataset because the BPE-dropout method did not surpass BPE, and we were constrained by computational resources.

We used truecase data during both training and inference. When we converted the generated output and reference to lowercase, the BLEU scores for BPE-dropout reached 26.20 and 32.50 for the En→De and De→En directions, respectively. These results are close to those reported in the BPE-dropout study [106], which used lowercase data for both training and inference. Although they pro-

vided results using truecase data for small datasets, they did not report the results using truecase data for large datasets. Moreover, our results indicated a negligible improvement in both directions of the WMT'14 En–De dataset when contrasting BPE-dropout with BPE. Provilkov et al. [106] reported similar results for a larger dataset with 16M lowercase training samples. We hypothesized that when using truecase data, the threshold for the amount of training data decreases. This is based on the result that improvement decreases when using truecase data, as shown in the Appendix of Provilkov et al. [106]. The BLEURT results are presented in Table 4.5, and the METEOR results are presented in Table 4.6, where a similar tendency to the BLEU results can be observed.

4.5.2 Comparison with Non-subword Methods

We evaluated DiverSeg against non-subword approaches on the IWSLT'15 En–Vi and WMT'16 En–Ro datasets. As presented in Table 4.7, the results indicate that the proposed method considerably outperforms non-subword approaches in terms of the BLEU scores. This finding is consistent with previous research, which has shown that subword methods such as BPE and SentencePiece often outperform character- or word-level approaches for source sentences [144].

	IWSLT'15 En–Vi		WMT'16 En–Ro	
	→	←	→	←
Character	27.04	24.72	29.82	30.49
Word	25.24	21.40	25.67	26.33
DiverSeg	**29.19**	**25.78**	**32.68**	**32.47**

Table 4.7: **Comparison to Non-subword Methods.** We report BLEU scores, and the best result is in **bold**.

4.5.3 Ablation Study of Proposed Modules

Table 4.8 present results of an ablation study assessing the effectiveness of the three proposed modules: 1) the lattice positional encoding (PE) that provides ab-

ID	Lattice PE	Subword Attn	$\mathcal{L}_{\text{Align}}$	BLEU	BLEURT	METEOR
1	✓	✓	✓	**29.19**	**56.20**	**33.78**
2	✓	✓	✗	28.81	55.44	33.51
3	✓	✗	✓	28.31	55.11	33.23
4	✓	✗	✗	28.45	54.40	33.49
5	✗	✓	✓	28.26	55.61	32.97
6	✗	✓	✗	27.52	54.11	32.77
7	✗	✗	✓	12.48	35.84	24.79
8	✗	✗	✗	13.85	37.64	26.04

Table 4.8: **Ablation of Components.** ✓ indicates using the component and ✗ indicates not using it. Tested on the En→Vi direction of the IWSLT'15 dataset and the best result is in **bold**. Default settings of DiverSeg are marked in purple.

solute positional information; 2) the subword-relation-aware attention mechanism that provides relative positional information; 3) the cross-granularity embedding alignment objective $\mathcal{L}_{\text{Align}}$. Comparison of ID 1 to 2, 3, and 5 indicates that each individual module contributes to the final result. Comparison of ID 5 and 6 reveals that $\mathcal{L}_{\text{Align}}$ is essential for the relative PE and significantly increases the BLEU score by 0.74. Additionally, at least one of the absolute position or the relative position is required (ID 7 and 8), otherwise, the model input degenerates into a bag-of-words input, leading to low performance. All ablation studies were conducted in the IWSLT'15 En→Vi direction, which is representative because of the moderate size of IWSLT'15.

4.5.4 Choice of Representing Subword Relations

We compared the choice of representing subword relations with previous methods: 1) Xiao et al. [148] did not utilize word information r_{ww}[5], 2) used *the* distance to represent the relative position between two subwords [118], and 3) did not use

[5]we use the remaining 18 types of relations instead of 7 types of relations in their paper

relative PE.

Table 4.9 shows that using relative information can substantially improve all three metrics compared to the model without relative PE. Moreover, using predefined relations like word boundary information and finer-grained relations (e.g., adjacency) in our proposed model outperformed the distance-based method proposed by Shaw et al. [118]. Word boundary information improves all three metrics.

	BLEU	BLEURT	METEOR
Proposed (Table 4.1)	**29.19**	**56.20**	**33.78**
Proposed w/o r_{ww}	29.18	56.15	33.64
Relative PE [118]	28.93	55.37	33.43
w/o relative PE	28.31	55.11	33.23

Table 4.9: **Ablation of Subword Relations.** Test on the En→Vi direction of the IWSLT'15 dataset.

4.5.5 Choice of Metrics and Weight Ratio in Alignment Loss

We examined the metrics and weight ratios used to align various segmentations of one word. In addition to maximizing the cosine similarity of the embeddings as expressed in Eq. (4.17), we attempted to minimize the L1 or L2 distance. We also examined various weight ratios as shown in Eq. (4.19).

The results of using various metrics are presented in Table 4.10, which reveals that minimizing the distance causes the training to fail. The reason for this is illustrated in Figure 4.4: **1)** During maximizing the cosine similarity, the norms of embeddings remain the same; **2)** however, while minimizing the L1 or L2 distance, the embeddings drift toward the origin because embeddings with small norms tend to have a small distance. We alleviated this problem by embedding normalization $\mathbf{E}_{\text{n}} = \frac{\mathbf{E}}{\text{norm}(\mathbf{E})}$ and labeled them as Norm L1 and Norm L2 distances. The normed versions exhibited superior performance; however, their performance was poorer than those using cosine similarity. This result can be attributed to the "curse

of dimensionality," where the distance between any two embeddings tends to be similar and their differences cannot be captured meaningfully [124].

	BLEU	BLEURT	METEOR
Cosine Similarity	**29.19**	**56.20**	**33.78**
L1 distance	0.81	20.30	10.97
L2 distance	21.67	45.28	30.23
Norm L1 distance	24.35	48.86	31.40
Norm L2 distance	28.57	55.61	33.69

Table 4.10: **Ablation of Similarity Measurements.** Test on IWSLT'15 En→Vi. The best result is in **bold**.

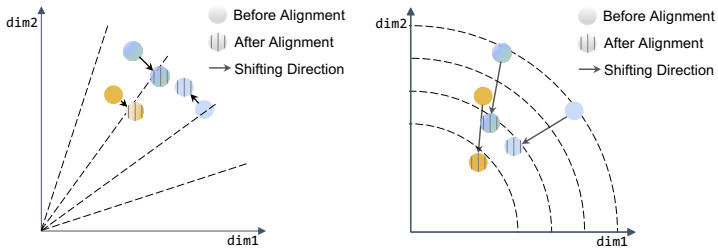


Figure 4.4: **Embedding Shifting Visualization.** The shifting of hidden representations by **left**: maximizing cosine similarity, and **right**: minimizing L2 distance in the cross-granularity alignment loss function.

The results of using various weight ratios are presented in Figure 4.5, showing ratio as 1 or 2 is the optimal setting for the IWSLT'15 En→Vi direction. We can observe that it is an unimodal function in terms of BLUE score where setting the weight ratio too low or too high negatively impacts performance.

4.5.6 Choice of Segmenter Combinations

Table 4.11 presents a comparison of the various combinations of segmenters. The combination of character-level, BPE, and BERTSeg delivered the highest translation performance, as indicated by the BLEU score within the *Non-stochastic*

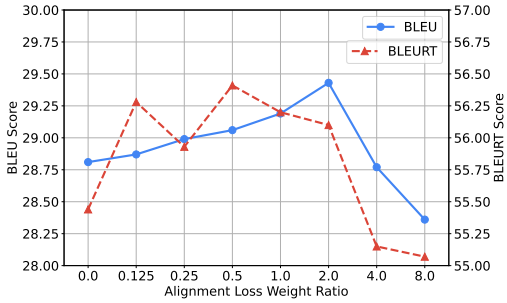


Figure 4.5: **Ablation of Alignment Loss Weight Ratios.** Test on IWSLT'15 En→Vi.

Method group. Furthermore, the combination of character-level and BPE-dropout outperformed the combination of character-level, BPE, and BERTSeg. Based on this finding, we suggest not combining dynamic segmentation with fixed segmentation because dynamic segmentation may cover fixed segmentation in certain epochs, except for character-level segmentation which provides positional information.

Moreover, the diversity of the segmenters in a combination is a crucial factor. We define the segmentation difference rate using the proportion of words in the corpus that are segmented differently by two segmenters, where each type of word counts only once. Figure 4.6 illustrates that the difference rates between BPE and BERTSeg, and between BPE and SentencePiece were high. Therefore, combining BPE and BERTSeg (SentencePiece) provides diverse perspectives of the word in the source language and a dense input lattice, resulting in excellent translation quality. We found a low difference rate between BPE and DPE, which may explain the low translation quality when combining them.

4.6 Analysis

In this section, we present several analyses that investigate the effectiveness of DiverSeg. We provide translated examples in Section 4.6.1, introduce a new metric called target word accuracy in Section 4.6.2, visualize the attention mechanism in Section 4.6.3, and visualize embedding in Section 4.6.4 to gain insights into how

	BLEU	BLEURT	METEOR
Char+BPE+BERTSeg	**29.19**	56.20	**33.78**
Char+BPE+DPE	28.56	55.88	33.17
Char+BPE+SentencePiece	28.98	**56.22**	33.57
Char+BERTSeg+DPE	28.84	55.74	33.64
Stochastic Method			
Char+BPE Dropout	**30.84**	**58.57**	**34.36**
Char+BPE Dropout+BERTSeg	30.29	57.86	34.12

Table 4.11: **Ablation of Segmenter Combinations.** Test on the En→Vi direction of the IWSLT'15 dataset. The best result among each group is in **bold**.

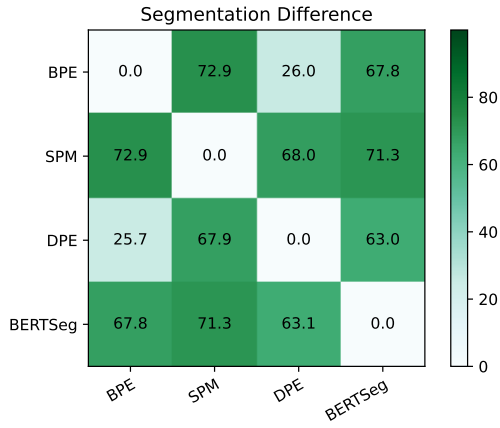


Figure 4.6: **Segmentation Difference Rate.** Numbers indicate the percentage of words where two segmenters provided distinct segmentations.

DiverSeg improves translation quality.

4.6.1 Translated Examples

We present an example in Table 4.12 and demonstrate how DiverSeg produces the correct translation as evidenced by the attention heatmap in Figure 4.7. In the example, the Vietnamese word "nói" should be translated into "speak" in En-

glish as in the reference. However, both BPE and BERTSeg produced incorrect translations “say” and “talk,” respectively. The translation generated by BERTSeg exhibited a higher semantic similarity with the reference, and only DiverSeg correctly translated the word. The attention heatmap in Figure 4.7 reveals that when generating the correct English word “speak,” DiverSeg attends to various segmentations of the corresponding Vietnamese word “tập nói” (learn to speak).

Method	Sentence	BLEURT
Source (Vi)	Ý tôi là, thử nghĩ mất bao lâu để một đứa trẻ tập nói.	-
Reference	I mean, just think how long it takes a child to learn to speak.	-
BPE	I mean, I think it’s going to take a child to say.	37.63
BERTSeg	I mean, let’s take a long time to get a kid talk.	44.98
DiverSeg	I mean, it took a long time for a child to speak.	**73.91**

Table 4.12: **Translation Example.** The proposed DiverSeg method correctly translates the source word to the target word “speak.” From the IWSLT’15 Vi–En dataset.

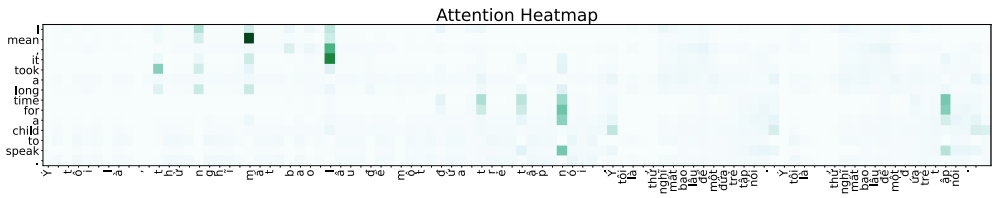


Figure 4.7: **Attention Heatmap.** In the DiverSeg model from the target sentence to the source sentence.

4.6.2 Target Word Accuracy with Different Frequency Ranges

Table 4.13 presents a comparison of the accuracy of translating the target words with varying frequencies. Accuracy was calculated by extracting each Vietnamese word from the reference sentence, determining its frequency in the training set, and evaluating whether it appeared in the translation. The proposed DiverSeg outper-

formed all other methods in almost all frequency bins, demonstrating its effectiveness and robustness. However, although the character-based NMT model achieved high accuracy for low-frequency words, primarily because of its copying ability, DiverSeg did not completely integrate this ability. Information from BERTSeg interferes with the generation process, whereas BERTSeg performs poorly on low-frequency target words but well on high-frequency words.

Frequency	[0, 10]	[11, 50]	[51, 100]	[101, 200]	[201, 500]	[501, 1000]	[1001, ∞)
Percentage	1.41%	1.77%	1.33%	2.36%	5.16%	6.66%	81.30%
Char	**31.76**	28.41	38.41	39.05	44.50	49.25	66.15
BPE	14.18	24.18	38.21	40.30	45.55	49.38	67.72
SentencePiece	7.37	26.53	37.60	41.18	46.76	52.16	67.85
VOLT	13.61	27.47	39.23	40.30	45.61	49.98	68.89
DPE	11.34	27.00	40.24	42.30	47.34	51.39	68.16
BERTSeg	8.88	27.00	39.23	42.43	48.45	52.68	68.38
DiverSeg	13.04	**31.55**	**44.72**	**44.31**	**52.03**	**54.48**	**70.04**

Table 4.13: **Word Translation Accuracy.** Translation accuracy comparison on target words of varying frequency evaluated in the IWSLT'15 En→Vi translation direction.

4.6.3 Visualization of Subword-relation-aware Attention

We demonstrated the roles of self-attention and subword-relation-aware attention through visualization using the BertViz tool [136]. The self-attention was extracted using Eq. (4.13) and subword-relation-aware attention was extracted using Eq. (4.14) from the last layer of the transformer encoder and averaged for all heads. As illustrated in Figure 4.8, in contrast to self-attention, which attends to all subwords, subword-relation-aware attention can overlook information from subwords belonging to multiple segmentations of the same word, indicating its ability to identify word boundaries.

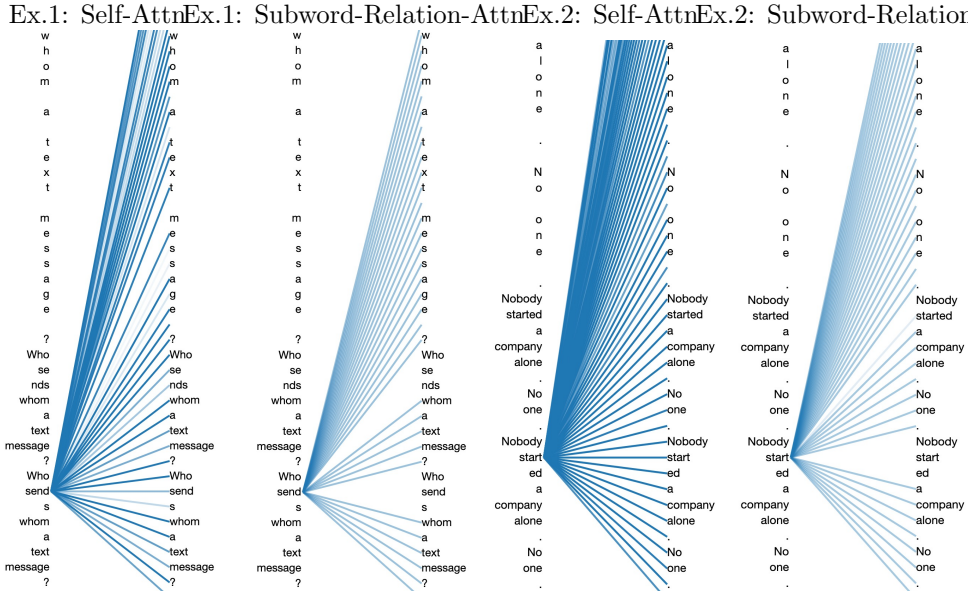


Figure 4.8: **Attention Visualization.** Two examples showing the difference between the self-attention mechanism and the proposed subword-relation-aware attention mechanism.

4.6.4 Visualization of Segmentation Embeddings

We investigated the effectiveness of the cross-granularity embedding alignment objective by comparing the word embeddings DiverSeg models with and without the alignment objective (ID 2 in Table 4.8). We generated the English word embeddings from a randomly selected sentence in the IWSLT'15 En→Vi direction during inference. Each point in Figure 4.9 represents the word embeddings calculated by averaging the embeddings of the subwords it contains. Subsequently, the subword embeddings were extracted from the final encoder layer. We only preserved words containing more than one BPE or BERTSeg subwords in the sentence and applied t-SNE [74] and principal component analysis (PCA) methods to map them to 2D points. The results revealed that with the alignment objective, different segmentations of a word tended to remain close, whereas without the alignment objective, the embeddings of the same word obtained by different segmentation methods were widely dispersed. For the t-SNE results, we observed

that with alignment, only three word embeddings calculated from character-level segmentation (Cameron, walking, and client) were not close to the word embeddings calculated from the corresponding BPE and BERTSeg segmentations. In the without-alignment method, word embeddings from character-level segmentation form an isolated cluster. This is more pronounced in PCA results, where, without alignment, embeddings from character-level segmentations are entirely distinct from other embeddings. This finding provides evidence that the proposed cross-granularity alignment objective successfully aligned the semantic meanings of multiple subword segmentations.

4.6.5 Efficiency

Time Efficiency. Methods that use a single segmenter take approximately 0.44 minutes to complete one epoch of training. This includes subword-based methods, such as BPE, SentencePiece, and BERTSeg. The character-based method takes 1.56 minutes for each epoch, primarily because of the longer input sequence. The method that employs multiple segmenters based on `Transformer Base` model [51] requires an average of 5.67 minutes to complete one epoch owning to the use of more data. The method of Xiao et al. [148] takes approximately 12.38 minutes,[6] while DiverSeg takes 16.56 minutes per epoch. This demonstrates a moderate efficiency reduction when compared to other methods that utilize multiple segmentations.

Parameters. The model parameters are listed in Table 4.14. DiverSeg is parameter efficient, where it uses only 6% more parameters compared with the model using BPE segmentation. Among the $2.3M$ extra parameters, only $0.2M$ parameters are from the subword-relation-aware attention module, and $2.1M$ parameters are from the larger look-up table in the encoder embedding layer because it covers both the BPE and BERTSeg vocabularies. Among the multi-segmenter methods, DiverSeg is almost as parameter efficient as the data mixing method [51]. In contrast, there are more parameters in Xiao et al. [148] because the look-up table covers more tokens in multiple BPE vocabularies, whose size is up to $16k$.

[6] This is presumably because the training speed of their model is 0.46 times that of the base model.

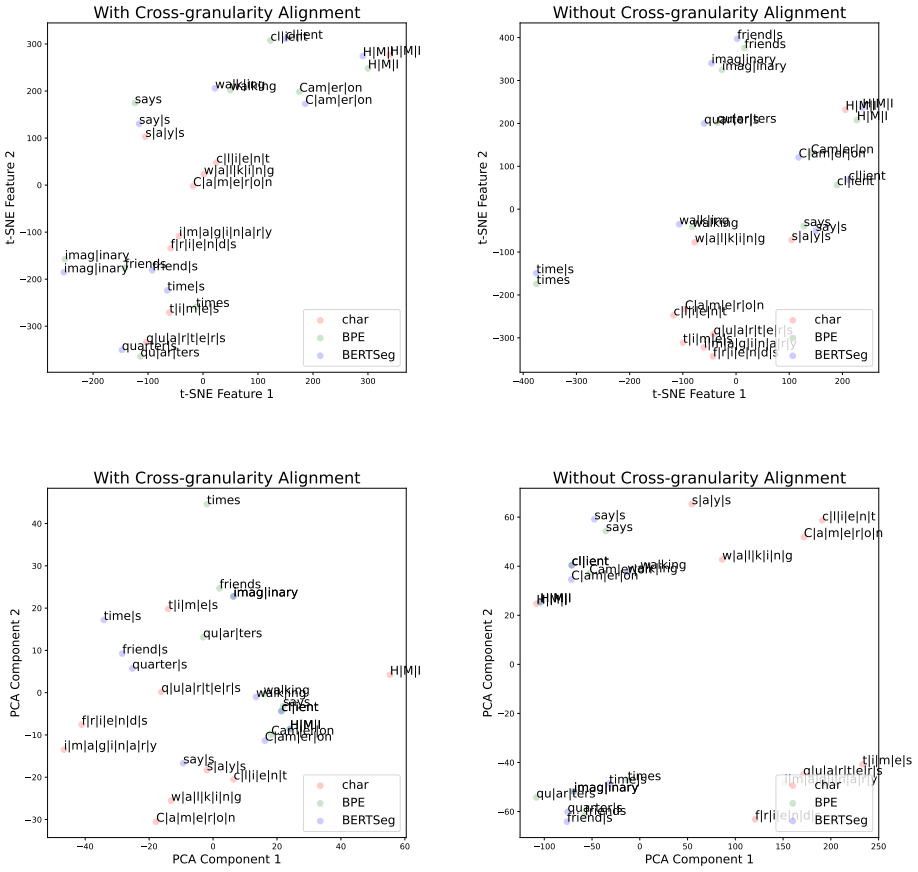


Figure 4.9: Word embedding visualization of one sentence using t-SNE (above) and PCA (below), where the left column shows embeddings with the cross-granularity alignment objective and the right column shows embeddings without the alignment objective. Word embedding is obtained by averaging all subword embeddings in that word, where | represents the subword boundary and colors represent the segmentation method for that embedding.

4.7 Conclusion and Future Work

In this study, we introduced DiverSeg to leverage diverse subword segmentations and capture the multiple perspectives of each input sentence. DiverSeg contains a lattice input module, a subword-relation-aware attention mechanism, and a cross-

Model	#Parameters
Single-Segmenter	
BPE	37.4M
SPM	37.5M
VOLT	35.9M
DPE	37.3M
BERTSeg	38.6M
Multiple-Segmenter	
Kambhatla et al. [51]	39.5M
Xiao et al. [148]	41.3M
DiverSeg	39.7M

Table 4.14: **Model parameters.** Obtained by actual measurement from models in the IWSLT'15 En→Vi translation direction.

granularity embedding alignment objective. We conducted extensive experiments on multiple machine translation datasets, and the results demonstrated that DiverSeg can be used to effectively exploit the potential of multiple segmentations to improve NMT performance. Ablation studies and analyses were performed to evaluate the contribution of each proposed module.

In the future, we will focus on designing a decoder architecture to enable learning from target sentences with multiple subword segmentations during training and exploring a decoding algorithm that combines the probabilities of multiple subword paths of a single word during inference. Additionally, the training speed is a limitation of DiverSeg. One potential method to expedite the training process is to optimize the computation of the cross-granularity loss.

Chapter 5

SubMerge: Merging Equivalent Subword Tokenizations for Subword Regularized Models in Neural Machine Translation

Despite the end-to-end nature that makes neural machine translation (NMT) [126, 4, 135, 33] the most prevalent and convenient approach for machine translation (MT), subword tokenization (or subword segmentation) [116, 106, 62, 61] remains an indispensable pre-processing step for most NMT systems. Subword vocabularies address the out-of-vocabulary problem of word-based NMT systems [50, 4, 73] by reducing new words to known subwords, while avoiding the high computational cost of character-based NMT systems [39, 53, 20, 69, 16] by enabling much shorter input and output sequences.

Deterministic segmenters like Byte-Pair Encoding (BPE) [116] and SentencePiece [61] are widely used due to their simplicity and effectiveness. They are determinisitic in the sense that they consistently generate the same tokenization for a given sentence. NMT models trained on consistent subword tokenizations typically allocate the majority of a sentence's true probability (considering all potential tokenizations by marginalizing over them) to its specific tokenization [12],

except for out-of-domain data [17]. Therefore, the probability of the sentence approximately equals the probability of the tokenization.

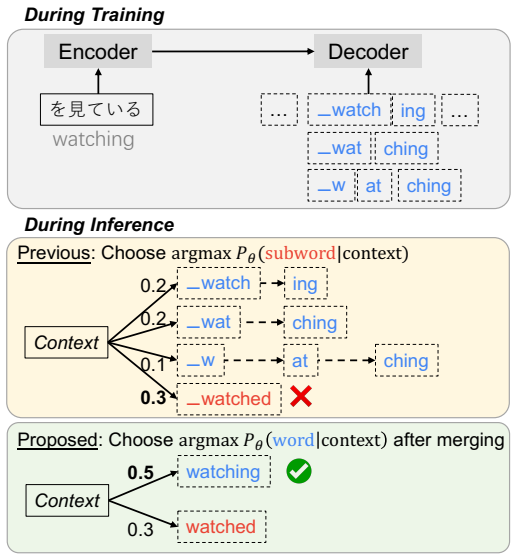


Figure 5.1: Subword regularized models suffer from discrepancies between training and inference, where they are trained on multiple target tokenizations and generate one. We propose to merge equivalent subword tokenizations that compose the same word with different conditional probabilities during the inference.

Subword regularization methods [106, 61] are stochastic segmenters - they produce multiple tokenizations of a given sentence during training, as shown in Figure 5.1. As a data augmentation method, models trained on regularized data usually outperform those trained on non-regularized data, especially in low-resource scenarios. However, this causes a discrepancy between training and inference. During training, the model learns to generate multiple target tokenizations for each source sentence. During inference, greedy or beam search approximates the single highest probability tokenization. This is the source of the discrepancy - the probability of a target tokenization diverges drastically from the probability of a target sentence marginalized over different tokenizations, because the model learns to distribute the probability of a target sentence across all the tokenizations it is exposed to during training. This inaccurate probability estimation of

the next word during inference leads to a degradation in translation quality. One way to overcome this is to incorporate the marginal likelihood of the next words during decoding for subword regularized models.

We propose SubMerge, a decoding algorithm that aggregates probabilities from exponentially many tokenizations for a sentence by merging equivalent subword tokenizations of words. The algorithm is based on the property of BPE-dropout [106] where each word is individually segmented. It is a nested beam search approach. In the outer beam search, we hide the detail of possible subword tokenizations of the word, treating words as minimal units. This ensures that the outer beam is unaware of and unaffected by the subword tokenizer. In the inner beam search, we limit the search space within the word boundary. The inner beam search finds the n-best tokenizations, merges equivalent ones, and returns a list of words and the corresponding probabilities.

Previous attempts to estimate marginal likelihood over tokenizations include summing over n-best tokenizations [12] and using importance sampling [17]. However, these algorithms focus on perplexity estimation, assuming the output is already in hand. In our approach, we perform marginal likelihood estimation for the next words during inference, aiming to improve not only the estimation precision but also the translation quality. In a nutshell, our contributions are as follows:
1. **We proposed SubMerge**, a nested beam search algorithm for generating text with subword regularized models. It merges equivalent subword tokenizations for the next words, thereby enhancing probability estimation accuracy and translation quality.
2. **Experimental results on five machine translation datasets** demonstrate significant improvements in estimating the underlying word perplexity computation for a model and its translation quality.
3. **We provide analyses** on using various beam sizes, the selection of the inner searching function, and the impact of hyperparameters.

5.1 Related Work

SubMerge is designed for decoding with text generation models for which likely tokenization probabilities diverge drastically from sentence probabilities. In other words, there are multiple tokenizations for one target sentence, and the probability distribution is splintered among them. Although we only experimented on models trained on data segmented by BPE-dropout [106], it also works for SentencePiece Regularization [61], MaxMatch-Dropout [43] and NMT models with multiple subword segmenters [51]. On the other hand, NMT models trained on sentences segmented by deterministic segmenter only benefit from marginal likelihood estimation in out-of-domain data or long words [12, 17]. Deterministic subword segmentation includes not only subword-level methods such as WordPiece [112], BPE [116], SentencePiece [62], Dynamic Programming Encoding [41], BERTSeg [121], but also byte-level [117], character-level [129], word-level [91], and hybrid word-character methods [72].

Marginal likelihood estimation can be implemented through two ways, sampling and dynamic programming. Sampling methods include summing over n-best tokenizations [12] or important tokenizations [17]. Sampling can be easily applied to any generation model. However, a manageable number of tokenizations cannot precisely estimate the probability of sentences with an exponentially large number of tokenizations, which is the case during the inference of the subword regularized models. On the other hand, dynamic programming can handle an exponentially large number of tokenizations by merging the same historical states, as introduced in *sequence modeling via segmentations* [143] and applied in the mixed-character-subword models [41, 89]. However, they merge the historical states by approximating the previous output by character-level data. That is, after the decoder generates one subword, it is split into characters and fed to the decoder. This is not applicable to pure subword models. Based on the property that each word is individually segmented in BPE-dropout [106], we obtain n-best tokenizations within a small search space and treat the best tokenization of each word the historical state, taking advantage of both marginal likelihood estimation methods.

5.2 Preliminaries

This section formulates the objective of the inference process for NMT models, highlights the distinction introduced by subword regularized models, and how we address it.

The inference objective of an NMT model with parameters θ is to obtain $\arg\max_Y P_\theta(Y|X)$ where X and Y are the source and target sentences in plain text form. For subword-based NMT models, we tokenize X into a sequence of tokens during both training and inference. We tokenize Y during the training and try to predict a sequence of tokens that compose Y during inference. We use two tokenizers $\tau_S(X) = \boldsymbol{x}$, where $\boldsymbol{x} = (x_1, ..., x_n)$ and $\tau_T(Y) = \boldsymbol{y}$, where $\boldsymbol{y} = (y_1, ..., y_m)$. Each subword x_i or y_i is a non-empty substring of the text X or Y in a finite-size subword vocabulary predefined by the source or target tokenizer. In theory,

$$P_\theta(Y|X) \neq P_\theta(\boldsymbol{y}|\boldsymbol{x}), \tag{5.1}$$

because there are multiple tokenizations of X and Y (besides $\boldsymbol{x}$ and $\boldsymbol{y}$) that the model P_θ would assign non-zero probabilities to [12].

For non-regularized models using deterministic tokenizers such as BPE [116], since $\tau(\cdot)$ is a bijective function, we can approximate the objective using one tokenization with a gap less than 0.5% [17]:

$$P_\theta(Y|X) \approx P_\theta(\boldsymbol{y}|\boldsymbol{x}). \tag{5.2}$$

Therefore, we can use $\arg\max_{\boldsymbol{y}} P_\theta(\boldsymbol{y}|\boldsymbol{x})$ to approximate $\arg\max_Y P_\theta(Y|X)$ with greedy or beam search in inference. This allows us to identify the next tokens with high conditional probabilities without concern for the discrepancy between the probability of raw text Y and of the particular tokenization $\boldsymbol{y}$.

For subword regularization methods [106], however, the tokenizer τ stochastically yields multiple tokenizations for one sentence. That is $\tau_S(X) = \boldsymbol{x} \in \mathcal{V}_\mathcal{S}(\mathcal{X})$ where $\boldsymbol{x} \sim P_{\tau_S}(\boldsymbol{x}|X)$. Similar for Y. In this case, the size of possible segmentation $\mathcal{V}_\mathcal{S}(\mathcal{X})$ increases exponentially according to the length of X, which deviates

$P_\theta(\boldsymbol{y}|\boldsymbol{x})$ drastically from $P_\theta(Y|X)$, thus it requires marginalization over all possible tokenizations:

$$P_\theta(Y|X) = \sum_{\boldsymbol{x}\in\mathcal{V}_S(X)} \sum_{\boldsymbol{y}\in\mathcal{V}_T(Y)} P_\theta(\boldsymbol{y}|\boldsymbol{x}) P_{\tau_S}(\boldsymbol{x}|X). \tag{5.3}$$

This study focuses on better estimating the marginal likelihood of the target side, so we simplify Eq. (5.3) by using the most probable source tokenization $\arg\max_{\boldsymbol{x}\in\mathcal{V}_S(X)} P_{\tau_S(\cdot)}(\boldsymbol{x}|X)$ and remove the effect of the source tokenizer, resulting in:

$$P_\theta(Y|X) \approx \sum_{\boldsymbol{y}\in\mathcal{V}_T(Y)} P_\theta(\boldsymbol{y}|\boldsymbol{x}). \tag{5.4}$$

We propose SubMerge to approximates Eq. (5.4) by introducing an intermediate variable, word tokenizations $\boldsymbol{w} = (w_1, ..., w_n)$, generated by a word tokenizer $\tau_W(\cdot)$ which is a bijective function.[1] The problem is simplified as:

$$P_\theta(Y|\boldsymbol{x}) = P_\theta(\boldsymbol{w}|\boldsymbol{x}) = \prod_{i=1}^{n} P_\theta(w_i|\boldsymbol{w}_{<i}, \boldsymbol{x}). \tag{5.5}$$

We estimate $P_\theta(w_i|\boldsymbol{w}_{<i}, \boldsymbol{x})$ by summing over probabilities of subword tokenizations for one word w_i where the search space is much smaller compared to the search space of tokenizations of a whole sentence in Eq. (5.4):

$$P_\theta(w_i|\boldsymbol{w}_{<i}, \boldsymbol{x}) \approx \sum_{\boldsymbol{y}'\in\mathcal{V}_T(w_i)} P_\theta(\boldsymbol{y}'|\boldsymbol{w}_{<i}, \boldsymbol{x}). \tag{5.6}$$

In practice, since the decoder only takes subword as input, we feed the best subword tokenization of the next word w_i, which is $\arg\max_{\boldsymbol{y}'\in\mathcal{V}_T(w_i)} P_\theta(\boldsymbol{y}'|\boldsymbol{w}_{<i})$.
In this way, the probability of the target sentence is accurately calculated through a deterministic word tokenization as shown in Eq. (5.5), where the probability estimation of each word is precisely estimated through marginal likelihood estimation shown in Eq. (5.6). We implement Eq. (5.5) with the outer beam search as introduced in Section 5.3.2 and Eq. (5.6) with our inner beam search as introduced in Section 5.3.3.

[1] That is $\tau_W(Y) = \boldsymbol{w}$. Note that word tokenizer is not a bijective function for languages such as Japanese or Chinese. For these languages, we can use specific word segmentaters such as Jumanpp or Stanford Word Segmenter, which are bijective.

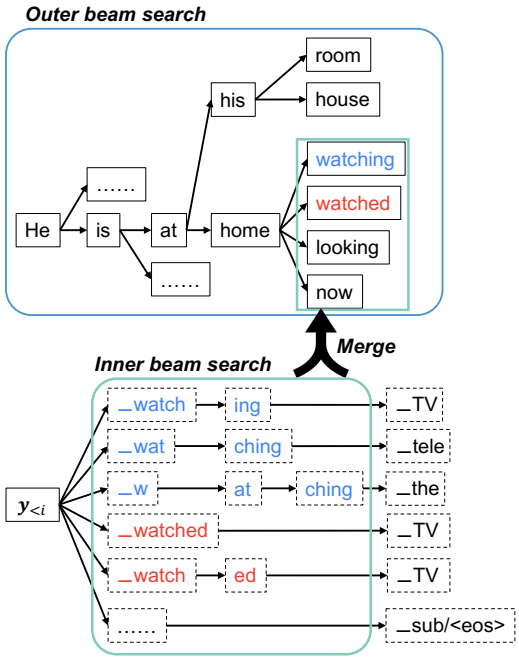


Figure 5.2: Overview of SubMerge. It contains an outer beam search that views words as minimal units. The candidate words and their probabilities are obtained from merging subword tokenizations in the n-best list of the inner beam search.

5.3 Methodology

5.3.1 Overview of SubMerge

An overview of the SubMerge algorithm is shown in Figure 5.2. It is a nested beam search decoding algorithm that contains an outer beam search as explained in Section 5.3.2 and an inner beam search with subword merging post-processing as explained in Section 5.3.3. The outer beam search selects from a list of words considering the conditional probability in each step and estimates the most probable sentence $\arg\max_Y P_\theta(Y|X)$. The inner beam estimates the conditional probability of words in Eq. (5.6) by merging the probabilities of different subword tokenizations of the same words.

5.3.2 Outer Beam Search

The outer beam search algorithm is shown in Algorithm 2. It follows the standard beam search approach, with words serving as the basic units. The candidates of the next words are obtained from a call to the inner beam search rather than the probability distribution generated by the decoder. Each state in the outer beam search queue contains the probability of the generation, the previous words, and their most probable tokens. Each state s' from the inner beam search contains the probability of the possible next word, the next word itself, and the most probable subword tokenization of that word.

In practice, we take the logarithm ($log(\cdot)$) of the probabilities for computational accuracy. We implemented early stopping after all sequences reach the special end-of-sentence (¡eos¿) token.

5.3.3 Inner Beam Search

The inner beam search is shown in Algorithm 3. It consists of two parts: a token-level beam search within the word boundary and post-processing to merge probabilities from equivalent subword tokenizations that compose the same word.

The inner search stops exploring a sequence of subwords when it reaches the start of the next word (with the start-of-word indicator '▁' Unicode U+2581) or the ¡eos¿ token. This stopping token will not be added to the token list. We remove special tokens and spaces during the detokenization of a token list to form the word and return a list of words with their probabilities. The time complexity of SubMerge is $O(T \cdot K^3)$, where T is the sentence length and K is the beam size. The derivation is presented in Appendix C.1.

5.4 Experimental Setup

We introduce the MT datasets used in our experiments, as well as our preprocessing settings, including subword tokenization in Section 5.4.1. In Section 5.4.2, we provide details around the training and inference of our models, and in Section 5.4.3, we present our evaluation setup.

Algorithm 2: OuterBeamSearch

Data: Beam width K, max length T

Result: Best sequence of states

Initialization:

$B_0 \leftarrow \{(0, [], [])\}$;

for $t \leftarrow 1$ **to** T **do**

  $B_t \leftarrow \emptyset$;

  foreach $s \in B_{t-1}$ **do**

    if s reaches $< eos >$ **then**

      $B_t.append(s)$;

      continue;

    foreach $s' \in \text{InnerBS}(s[2])$ **do**

      $score, word, toks = s'$;

      $score \leftarrow s[0] + score$;

      $words \leftarrow s[1] + words$;

      $toks \leftarrow s[2] + toks$;

      $B_t.append((score, words, toks))$;

  Sort B_t by scores in descending order.;

  $B_t \leftarrow B_t[: K]$

return B_T

Algorithm 3: InnerBeamSearch

Data: Beam width K, max length T, $toks$

Result: Next word list

Initialization:

$B_0 \leftarrow \{(0, toks)\}$;

for $t \leftarrow 1$ **to** T **do**

  $B_t \leftarrow \emptyset$;

  foreach $s \in B_{t-1}$ **do**

    if s reaches _ or $< eos >$ **then**

      $B_t.append(s)$;

      continue;

    foreach $s' \in Decoder(s[1])$ **do**

      $score, toks = s'$;

      $score \leftarrow s[0] + score$;

      $toks \leftarrow s[1] + toks$;

      $B_t.append((score, toks))$;

  Sort B_t by scores in descending order.;

  $B_t \leftarrow B_t[: K]$

$W = \{\}$;

foreach $s \in B_T$ **do**

  $score, toks = s$;

  $word = detokenize(toks)$;

  if $word \notin W$ **then**

    $W[word] = (score, toks)$

  else

    $W[word][0] += score$

return $list(W.items())$

5.4.1 Data and Pre-processing

Dataset	Train	Valid	Test
ALT Asian Langs–En	$18k$	$1,000$	$1,018$
IWSLT'15 Vi–En	$133k$	$1,553$	$1,268$
WMT'16 Ro–En	$612k$	$1,999$	$1,999$
WMT'15 Fi–En	$1.8M$	$1,500$	$1,370$
WMT'14 De–En	$4.5M$	$45,781$	$3,003$

Table 5.1: Statistics of the datasets.

Datasets We conducted experiments in low- to high-resource MT settings with datasets listed in Table 5.1, including Asian Language Treebank (ALT), IWSLT'15 English–Vietnamese (En–Vi), WMT'16 English–Romanian (En–Ro), WMT'15 English–Finnish (En–Fi), and WMT'14 English–German (En–De) datasets. ALT is a multi-way parallel dataset containing data in English and other Asian languages including Filipino (Fil), Indonesian (Id), Japanese (Ja), Malay (Ms), Vietnamese (Vi), and simplified Chinese (Zh). We used the ALT-standard-split tool[2] to split the dataset into train, validation, and test sets.

Data Pre-processing We performed word tokenization on all data. We applied Juman++ [132] to data in Japanese, Stanford-tokenizer [75] to data in Chinese, and Moses tokenizer [58] to data in other languages. We normalized Romanian data and removed diacritics following previous work [115]. We prepared the WMT'14 English–German dataset using a data cleaning and normalization tool from Fairseq.[3]

We applied subword tokenization to each translation direction separately. For source or target language, we trained a subword tokenizer with a subword vo-

[2] `www2.nict.go.jp/astrec-att/member/mutiyama/ALT`

[3] `github.com/facebookresearch/fairseq/blob/main/examples/translation/prepare-wmt14en2de.sh`

cabulary of $8k$ on the monolingual corpus. We applied a widely adopted toolkit[4] to train BPE-dropout tokenizers with a dropout rate of 0.2 for the generation of subword-regularized data and train BPE tokenizers for the generation of non-regularized data.

5.4.2 NMT Settings

Model We conducted MT experiments using the Fairseq framework [100] with the base architecture of the Transformer model [135]. We set dropout and attention dropout rates to 0.1. We applied layer normalization [64] for both the encoder and decoder.

Training We set the batch size to $3,072$ tokens in the source language and used eight GPUs, resulting in 25k source tokens per batch. We used the Adam optimizer [54] with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. We used warmup and linear decay for the learning rate following Vaswani et al. [135], with 4k warm-up steps, an initial learning rate of $1.7 * 10^{-7}$ and a final learning rate of 5×10^{-4}. We used label smoothing for the cross entropy loss with $\epsilon_{ls} = 0.1$ [127]. We calculated the loss on the validation set after each epoch and applied early stopping when no improvement was observed for 10 epochs.

Inference We selected the checkpoint with the best loss on the validation set. We used beam search and SubMerge with a beam size of 4 without additional normalization techniques, such as length penalty or temperature sampling [27].

5.4.3 Evaluation Metrics

We report word perplexity on generated translations to compare the probabilities assigned to generations by models. To evaluate translation quality, we report BLEU using sacreBLEU [105][5] and chrF++ [104][6], performing paired bootstrap resampling for statistical significance tests [56].

[4] `github.com/google/sentencepiece`

[5] BLEU+c.mixed+l.en-lang+#.1+s.exp+tok.13a+v.1.5.1

[6] `github.com/m-popovic/chrF` with c6w2F0.4

The word perplexity is calculated as follows. We first evaluate the negative log probability of the generated sentences for models using SubMerge by:

$$s_{score} = -\sum_{i} \log P_{\theta}(word_i), \tag{5.7}$$

and models with beam search by:

$$s_{score} = -\sum_{i} \log P_{\theta}(tok_i). \tag{5.8}$$

We evaluated the word perplexity by

$$w_{ppl} = exp(\frac{1}{N} s_{score}), \tag{5.9}$$

where N is the number of words. We evaluate the word perplexity on the generated hypothesis, rather than on the reference. This reflects the actual scenario in generation tasks where we dynamically generate the next token (word) conditioned on what the model has generated instead of on the ground truth. Nevertheless, word perplexity is a conditional probability that is dependent on not only the input but also the parameters in the model. Therefore, the perplexity results always need to be considered along with model-independent metrics such as BLEU scores.

5.5 Main Results

The results for subword regularized models are shown in Table 5.2. SubMerge led to better word-level perplexities than traditional beam search and higher BLEU and chrF++ scores, often achieving statistically significant improvements.

Word perplexity results improved substantially in the regularized models in contrast to the tiny gap (0.5%) reported in the non-regularized models [17] and in our analysis shown in Section 5.6.5. This is due to the fact that multiple tokenizations for one word appeared during training, which acts as a label-smoothing function on multiple correct next tokens. Therefore, the probability weight is distributed across multiple subwords thus, it becomes necessary to incorporate the marginal likelihood. It is worth noting that here word perplexity represents the accuracy of probability estimation rather than fluency or quality of the output.

	Word Perplexity ↓		BLEU ↑		chrF++↑	
	Beam Search	SubMerge	Beam Search	SubMerge	Beam Search	SubMerge
Low-resource Scenario						
ALT Fil→En	12.68	**4.59**	31.10	**31.82*** $_{+0.7}$	57.98	**59.17**$_{+1.2}$
ALT En→Fil	9.56	**4.14**	30.20	**31.14*** $_{+0.9}$	59.64	**60.14**$_{+0.5}$
ALT Id→En	17.91	**5.91**	27.35	**28.73*** $_{+1.4}$	53.61	**56.39**$_{+2.8}$
ALT En→Id	16.44	**4.91**	33.63	**34.19** $_{+0.6}$	63.14	**63.89**$_{+0.8}$
ALT Ja→En	24.90	**7.79**	15.07	**15.26*** $_{+0.2}$	45.07	**45.46**$_{+0.4}$
ALT En→Ja	6.55	**3.69**	14.38	**14.59** $_{+0.2}$	27.92	**29.02**$_{+1.1}$
ALT Ms→En	11.28	**4.33**	31.86	**32.16*** $_{+0.3}$	59.01	**60.09**$_{+1.1}$
ALT En→Ms	12.82	**4.18**	38.83	**39.28** $_{+0.5}$	66.25	**66.91**$_{+0.7}$
ALT Vi→En	17.21	**6.14**	23.64	**24.97*** $_{+1.3}$	52.32	**52.93**$_{+0.6}$
ALT En→Vi	8.64	**3.52**	27.35	**27.64** $_{+0.3}$	53.66	**53.82**$_{+0.2}$
ALT Zh→En	23.11	**7.81**	13.92	**14.31*** $_{+0.4}$	43.54	**44.43**$_{+0.9}$
ALT En→Zh	13.61	**6.76**	9.03	**9.87*** $_{+0.8}$	22.76	**23.25**$_{+0.5}$
Middle- and High- Resource Scenario						
IWSLT'15 Vi→En	14.41	**5.62**	27.87	**28.43*** $_{+0.6}$	48.62	**50.59**$_{+2.0}$
IWSLT'15 En→Vi	7.98	**3.39**	28.08	**28.16** $_{+0.1}$	49.27	**50.18**$_{+0.9}$
WMT'16 Ro→En	7.44	**3.22**	**33.85**	33.77 $_{-0.1}$	58.75	**59.07**$_{+0.3}$
WMT'16 En→Ro	6.78	**3.11**	34.35	**34.50** $_{+0.1}$	58.66	**58.89**$_{+0.2}$
WMT'15 Fi→En	11.27	**4.27**	**18.95**	18.88 $_{-0.1}$	47.24	**47.55**$_{+0.3}$
WMT'15 En→Fi	22.52	**7.81**	16.51	**16.65** $_{+0.1}$	47.66	**47.97**$_{+0.3}$
WMT'14 De→En	10.33	**3.90**	28.85	**28.94** $_{+0.1}$	55.99	**56.52**$_{+0.5}$
WMT'14 En→De	12.74	**4.64**	24.69	**24.83** $_{+0.1}$	52.68	**52.77**$_{+0.1}$

Table 5.2: **Results of Subword Regularized Models.** Statistical significance $p < 0.01$ is indicated by * against Beam Search. SubMerge consistently improves over the Beam Search baseline in most directions. Word perplexity results represent the ability to accurately estimate sentence probability rather than fluency.

Translation quality is also improved, especially in low-resource scenarios where the average BLEU score improvement is 0.6, whereas in the middle- to high-resource scenarios, it is 0.3. We also observed consistent improvement in the chrF++ score. While only one translation direction among middle- and high-

resource directions is statistically significant, 8 out of 12 low-resource directions see statistically significant improvements.

5.6 Analysis

We investigate the effect of different beam sizes on the algorithm in Section 5.6.1. Section 5.6.2 explores using a sampling algorithm as the inner search algorithm. Section 5.6.3 and Section 5.6.4 respectively analyze the impact of the training set size and the dropout rate. Section 5.6.5 show conditions in which SubMerge is effective.

5.6.1 Assessing Beam Sizes Variants

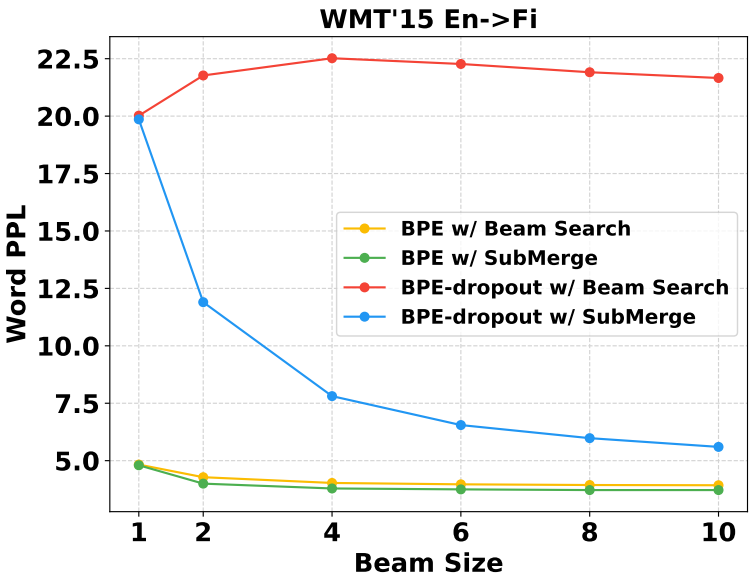


Figure 5.3: Word perplexity results using different beam sizes on the WMT'15 En→Fi direction.

Figures 5.3 and 5.4 show the word perplexities and BLEU scores of using different beam sizes for both non-regularized models and subword regularized models, comparing beam search and SubMerge. We observed that as we increased

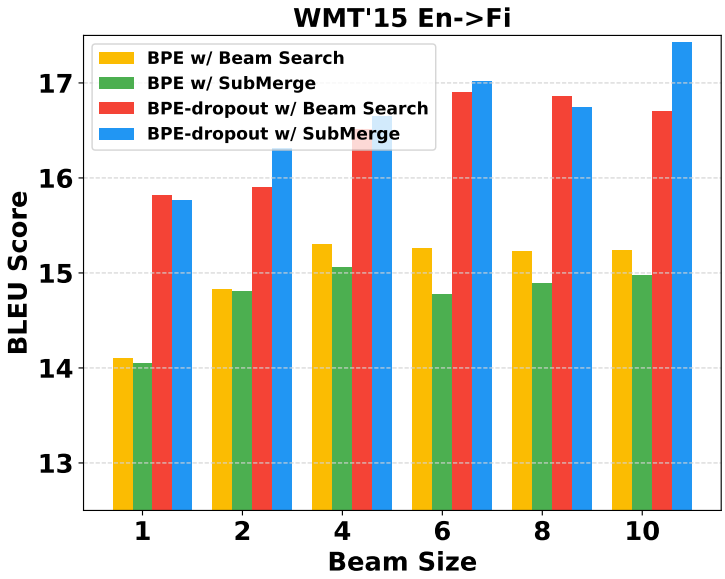


Figure 5.4: BLEU results using different beam sizes on the WMT'15 En→Fi direction.

the beam size, the word perplexity dropped sharply for BPE-dropout with SubMerge. When using a large beam size such as 10, it achieved comparable results to non-regularized models trained on one-best tokenization. Nevertheless, SubMerge does not yet accumulate as large a proportion of the probability distribution as using a non-regularized model. Since the training is on multiple segmentations, it certainly comes closer than when using beam search. For non-regularized models, combining equivalent paths for perplexity estimation also proved to be effective. We also observed that increasing beam size can lead to translation quality improvement for the SubMerge method. However, this is not the case for all directions as reported in Cohen and Beck [19].

5.6.2 Inner Search Algorithm Variants

We replaced the inner beam search with the sampling algorithm as shown in Appendix C.2 and compared word perplexity results in Figure 5.5. For the sampling algorithm, we sampled n^2 tokenizations (where n is the beam size) in the inner loop

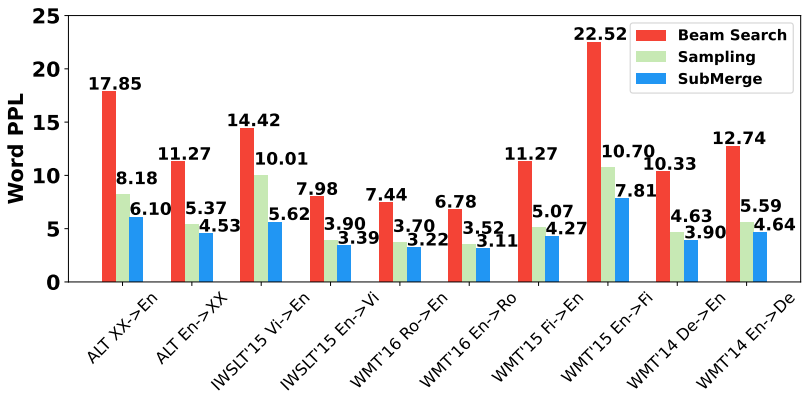


Figure 5.5: Word perplexity results comparing BPE-dropout with beam search to two variants of SubMerge - using either sampling as the inner search function or beam search as before.

and for each path, we started with the same historical information and selected the next subword according to the probability distribution until we reached the beginning of the next word. We then perform the same merging post-processing. However, we observed that the perplexity was higher than n-best tokenizations. This is because the sampling process could easily get lost at some step by selecting a token in the long tail with a very low probability.

5.6.3 Assessing Training Set Sizes

SubMerge is effective in extremely low-resource scenarios, as shown in Figure 5.6. We report BLEU scores using beam search and SubMerge during decoding for models trained on 1k to 18k parallel sentences. SubMerge consistently outperformed beam search across training set sizes. Moreover, the BLEU improvement reached approximately 3.4 using only 1k data. This observation reveals the potential of SubMerge to be used in domain adaptation scenarios with limited data.

5.6.4 Impact of Dropout Rates

Using a smaller dropout rate in BPE-dropout yielded lower word perplexity and higher BLEU scores in high-resource scenarios, as shown in Table 5.3. When the

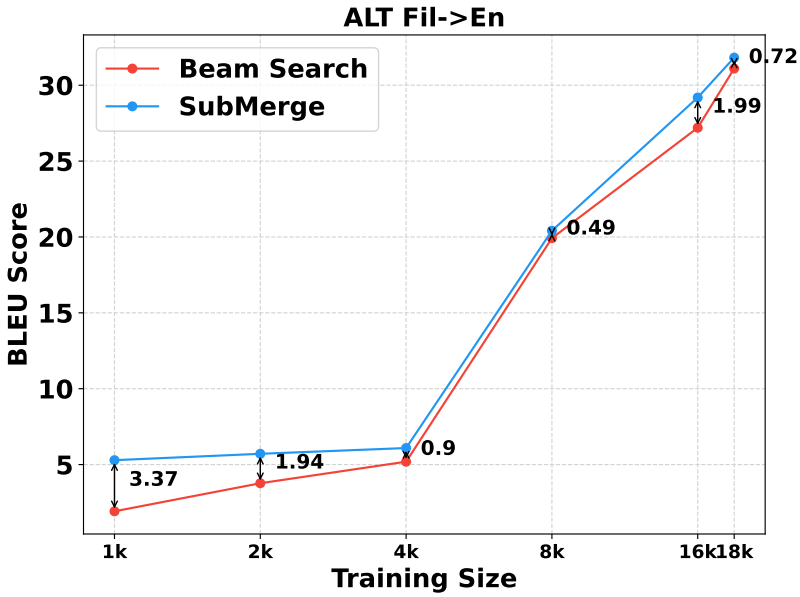


Figure 5.6: Translation quality using different sizes of training data. The x-axis is logarithmized.

dropout rate is low, the randomness of subword segmentation for a given word also decreases, leading to reduced variability in the training data and, concurrently, a diminished range of choices during the inference process. In the context of low-resource scenarios, reduced variability implies diminished data augmentation, which can adversely affect the model's generalization capability. Conversely, in high-resource settings, decreased variability signifies reduced noise, potentially enhancing model performance.

5.6.5 Does SubMerge Work on Non-regularized Models?

In short, No. We explored whether the proposed SubMerge method is applicable to non-regularized models using deterministic BPE tokenization. Table 5.4 presents word perplexities and BLEU scores on non-regularized models using beam search or SubMerge as the decoding algorithm.

We observed lower word perplexity using SubMerge compared to using beam search. However, the improvement is not as significant (approximately 6%) as

	Word PPL ↓		BLEU ↑	
Dropout Rate	0.1	0.2	0.1	0.2
ALT Others→En	**4.69**	6.10	22.06	**24.54**
ALT En→Others	**4.16**	4.53	24.75	**26.12**
IWSLT'15 Vi→En	**3.09**	5.62	**30.03**	28.43
IWSLT'15 En→Vi	**2.56**	3.39	**29.61**	28.16
WMT'16 Ro→En	**2.34**	3.22	**34.75**	33.77
WMT'16 En→Ro	**2.21**	3.11	**35.39**	34.50
WMT'15 Fi→En	**3.25**	4.27	18.87	**18.88**
WMT'15 En→Fi	**4.94**	7.81	16.64	**16.65**
WMT'14 De→En	**2.86**	3.90	**29.70**	28.94
WMT'14 En→De	**3.15**	4.64	**24.94**	24.83

Table 5.3: Results of SubMerge for models trained on BPE-dropout data with different dropout rates.

the improvement achieved by SubMerge for subword regularized models. This is consistent with our expectations. The model is trained on a single tokenization for each training word, so there is only one tokenization that accumulates most of the probability weight. For the non-regularized model, results show the translation quality of SubMerge is not as good as beam search. Therefore, the proposed SubMerge method is only applicable to subword regularized models in the NMT task.

For other tasks, such as question answering, the word perplexity is greater because the task is less structured than MT, where the source sentence is a highly limiting constraint. For less constrained tasks, it is possible that SubMerge will improve the performance of even non-regularized models. We leave this for future work to explore.

	Word PPL ↓		BLEU ↑	
	BeamSearch	Ours	BeamSearch	Ours
ALT Others→En	6.02	**5.60**	**15.73**	15.40
ALT En→Others	4.90	**4.77**	**18.06**	17.82
IWSLT'15 Vi→En	2.95	**2.79**	24.34	**25.63**
IWSLT'15 En→Vi	2.43	**2.42**	**25.09**	24.86
WMT'16 Ro→En	2.14	**2.11**	**32.05**	31.70
WMT'16 En→Ro	2.00	**1.98**	**32.98**	32.85
WMT'15 Fi→En	2.85	**2.76**	**17.08**	16.94
WMT'15 En→Fi	4.03	**3.79**	**15.30**	15.06
WMT'14 De→En	**2.39**	2.40	**30.18**	30.04
WMT'14 En→De	2.45	**2.36**	**25.88**	25.71

Table 5.4: **Results of Non-regularized Models.** We show the averaged results in En→ XX and XX→ En directions for the ALT dataset.

5.7 Conclusion and Future Work

We propose SubMerge to estimate the marginal likelihood of the next word by merging equivalent subword tokenizations during the inference of subword regularized models. Results demonstrate a significant improvement in word perplexity estimation and translation quality improvement in terms of BLEU and chrF++ scores, especially in low-resource scenarios.

Current inference algorithms are mostly based on conditional probability, which is a short-term value function. For future work of inference, we suggest aligning the value function towards evaluation metrics and human preference through reinforcement learning, where models are more aware of longer-term rewards.

5.8 Limitations

We did not experiment with common techniques in the beam search and SubMerge, such as length penalty. This is because we use a nested beam search, and the way to define the length (whether to use the number of tokens or the number of words) may differ from the definition in a traditional beam search. However, combining SubMerge with such techniques could be valuable for further work. Additionally, the implementation of beam search in Fairseq is not the standard one but an optimized one. For example, it uses twice the beam size during searching to prevent half of the candidates from reaching the end. As a result, setting the beam size to 1 should make beam search identical to greedy search, but it shows a better result as shown in Figure 5.4.

The word perplexity results reported in this paper are on the generated texts rather than on reference texts. They do not correlate with fluency or translation quality, and we only use them to report how much of the probability weight of a model is being used during decoding, which is still useful.

We use the SentencePiece tool for the current implementation of BPE and BPE-dropout algorithms. Therefore, the SubMerge implementation is also based on the format of this specific tool, which uses "▁" (U+2581) to represent the beginning of a new word. However, other tools may use "`@@`" at the end of a subword to indicate that the current word has not ended yet. Therefore, the implementation of SubMerge may be slightly different in terms of ending conditions in the inner beam search.

We trained NMT models by ourselves without any hyperparameter fine-tuning because we did not find any available checkpoints of subword regularized NMT models. There is a small gap between our results and the SOTA results. However, we believe the observations still hold for models with fine-tuned hyperparameters.

Chapter 6

Conclusion

This chapter first summarizes the thesis and provides the best practice. Then, we provide future directions for building a universal subword segmenter and using it in large language models.

6.1 Summary

In this thesis, we propose methods to address three challenges in the subword-based neural machine translation system in the segmentation, encoding, and decoding phases. The proposed methods improve the machine translation quality especially in low-resource scenarios.

Chapters 2 and 3 provide solutions for the optimal segmentation question. In Chapter 2, we propose SelfSeg, a fast neural subword segmenter based on masked pre-training. It defines the optimal segmentation(s) by the generation probability. During training, it maximizes the probabilities of all possible segmentations while during inference it yields one or multiple optimal segmentations according to their probabilities. Therefore, when the subword segmentations are used on the target side, they are easy for the NMT models to generate which results in improved translation quality. Moreover, it relies on monolingual word-level data, making it applicable to low-resource languages without large-scale parallel resources. In Chapter 3, we propose a BERT-based subword segmenter that generates subword segmentation that utilizes the contextualized semantic embeddings of words from

the BERT model. The segmentations better capture the semantic meaning of the word and show better performance on low-resource scenarios. Moreover, the pre-trained BERT encoder enables faster training than SelfSeg.

Chapter 4 provides solutions to encoding multiple subword segmentations of each word in the source language. In DiverSeg, multiple segmentations are encoded using a subword lattice input, a subword-relation-aware attention mechanism integrates relations among subwords, and a cross-granularity embedding alignment objective enhances the similarity across different segmentations of a word. We found incorporating information from multiple aspects enhances the performance of NMT, especially in low-resource scenarios.

In Chapter 5, we propose SubMerge, an algorithm that improves the word-probability estimation accuracy during decoding. It merges the probabilities of multiple subword segmentations that form the same word, which we call equivalent segmentations. This is specially designed for subword regularized NMT models, which leverage multiple subword segmentations of one target sentence during training. SubMerge is a nested search algorithm where the outer beam search treats the word as the minimal unit, and the inner beam search provides a list of word candidates and their probabilities, merging subword segmentations that form the equivalent word. It estimates the probability of the next word more precisely, providing better guidance during inference. We show it consistently outperforms the beam search algorithm in several machine translation datasets.

As for the best practice, in low-resource scenarios, we suggest to actively using data augmentation techniques including DiverSeg in the source side, BERTSeg-regularization in the target side, and SubMerge during decoding. In high-resource scenarios, considering the high computational cost and marginal improvement, applying SelfSeg-regularization on the target side is preferred.

6.2 Future Prospects

6.2.1 Universal Subword Segmenter

In this thesis, we train one subword segmenter for one particular language. However, a universal subword segmenter that can handle hundreds of languages will

be more practical in the application of multilingual machine translation systems where input and output sentences are possible to be any of the given languages.

Moreover, though BERT-based subword segmenter showed higher training and decoding speeds compared to previous neural segmenters, the speeds are still not satisfactory, given that frequency-based methods can process millions of sentences in seconds. Utilizing a look-up table or lightweight neural architecture are possible ways to achieve faster speeds.

6.2.2 Subwords in Large Language Models

In this thesis, the proposed methods enhanced translation quality especially for low-resource languages. LLMs, considered as universal solutions for NLP tasks, handle more than one hundred languages and many of them are low-resource ones. Since LLMs also use subwords as the minimal input and output unit and face the same challenges brought by subwords, the proposed methods also possible to improve the performance of LLMs when the questions or answers are in low-resource languages.

Appendix A

Supplementary Materials of BERTSeg

A.1 Limitations

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

Dependency on BPE Vocabulary BERTSeg is a model to learn optimal segmentations for words but not paired with a vocabulary generation algorithm. Currently, the vocabulary is generated by BPE, therefore, many subwords in the vocabulary are not used, as shown in Figure 3.2. It is possible to address this by first generating a large vocabulary and then shrinking it iteratively, saving the commonly used subwords only, motivated by the SentencePiece work [62].

Target Side Only The goal of BERTSeg is to maximize the generation probability as shown in Eq. (2.2), therefore, can only apply to the target side data in generation tasks. Applying BERTSeg to the source side data will not improve the MT performance in our preliminary experiments, which is also reported in the DPE work [41]. To address this, a dual segmenter model is needed to optimize both the target segmentations and source segmentations.

English Subword Segmenter Only Currently we only train the subword segmenter for English due to there is only an English characterBERT model. However, we believe using embeddings from BERT or mBERT will not affect the performance, although it adds a dependency on the BERT tokenizer. To extend BERTSeg to mBERTSeg, a multilingual characterBERT is needed. This will especially help low-resource or multilingual MT and NLP tasks [87, 80, 76, 79, 81, 85] where high-quality segmenters are not available.

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

A.2 Example: Segmentations

We provide examples comparing the proposed method with BPE including high-frequency words, rare words and unseen words as shown in Table A.1. We have the following observations:

- **For frequent words**, BERTSeg sometimes segment them into subwords even the word is in the vocabulary such as *official/s* and *use/d*. Additionally, the model can discriminate the ambiguous situations very well. For example, the model can extract the prototype *challenge* from the word *challenged*.

- **For rare words** with frequency < 5 in the training set, BERTSeg gives much better segmentations than BPE, because BPE is a frequency-based method and thus handles rare words poorly.

- **For unseen words**, although the BERTSeg model gives better segmentations than BPE, we found that sometimes it oversegments words such as *M/a/d/a/m/e*. We guess it's due to the low-quality word embedding from characterBERT, and we do not know the impact of this on the MT results.

A.3 Implementation Details of Baselines

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

A.3.1 BPE

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

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

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

A.3.2 VOLT

Table A.2 illustrates the optimal sizes of BPE vocabularies of each dataset calculated by the VOLT algorithm. The optimal numbers are very similar to the results we got from hyperparameter searching, showing the effectiveness of the VOLT algorithm.

A.3.3 BPE-dropout

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

A.3.4 DPE

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

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

Table A.1: BERTSeg and BPE segmentations on frequent words, rare words and unseen words.

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

Table A.2: Optimal BPE vocabulary sizes of languages in each dataset.

Appendix B

Supplementary Materials of DiverSeg

B.1 Vocabulary Size Selection

The optimal vocabulary sizes calculated by the VOLT algorithm [150] are presented in Table B.1.

Dataset	Language	Sizes
ALT	Asian Langs/En	[6k, 9k]/8k
IWSLT'15 Vi–En	Vi/En	7k/7k
WMT'16 Ro–En	Ro/En	11k/10k
WMT'15 Fi–En	Fi/En	8k/10k
WMT'14 De–En	De/En	11k/11k

Table B.1: Optimal BPE vocabulary sizes of languages in each dataset.

B.2 Detailed Results of MultiSub

During training, MultiSub [51] leverages character-level data, BPE segmentation, and BERTSeg segmentation in the *English→the other language* direction and character-level data, BPE segmentation, and SentencePiece segmentation in *the other language→English* direction. During inference, we test the performance using the sentence segmented by every single segmenter. As presented in Table B.2, using the character-level input data achieved a very low performance, whereas using BPE or BERTSeg improved performance.

DiverSeg enhanced the connection between the three segmentations through the subword-relation-aware attention mechanism in the model and the cross-granularity embedding alignment during the training process. Therefore, the performance was considerably higher than using the same three segmenters.

	En–Fil		En–Id		En–Ja		En–Ms		En–Vi		En–Zh		Avg	
	→	←	→	←	→	←	→	←	→	←	→	←	→	←
MultiSub														
Char	5.54	3.82	4.94	5.19	2.55	4.37	7.41	7.24	3.73	6.87	1.02	5.84	4.20	5.56
BPE	17.81	14.53	20.51	17.41	9.77	6.31	24.83	19.59	19.94	13.42	5.30	7.29	16.36	13.09
BERTSeg/SPM	17.28	14.83	19.52	16.89	9.58	6.06	24.14	18.89	19.20	12.09	5.21	7.74	15.82	12.75
Avg	13.54	11.06	14.99	13.16	7.30	5.58	18.79	15.24	14.29	10.79	3.84	6.96	12.13	10.47
DiverSeg	22.89	18.28	25.22	20.80	10.72	7.73	30.51	26.28	24.02	16.49	8.60	8.52	20.33	16.35

Table B.2: BLEU results on the ALT dataset using input by each segmenter in the MultiSub method.

Appendix C

Supplementary Materials of SubMerge

C.1 Derivation Process for Time Complexity

In the Outer Beam Search Algorithm 2, the loop from Line 3 contains at most T steps, and the loop in Line 5 contains at most K steps because we limited the size in Line 16. Therefore, the time complexity of the outer beam search is $O(T \cdot K) \cdot O(InnerBeamSearch()))$.

In the Inner Beam Search Algorithm 3, the loop from Line 3 contains at most T steps, the loop from Line 5 contains at most K steps and the loop from Line 9 contains at most K steps because each beam yields at most K candidates by selecting tokens with top-K highest probability. Therefore, the time complexity of the inner beam search is $O(T \cdot K \cdot K)$.

Since the max length limitation T is actually performed on the sentence level, where $T >= \sum_i |w_i|$, we do not need to count T twice. The overall time complexity is $O(T \cdot K^3)$.

C.2 Sampling as Inner Seach Function

We can replace the inner beam search algorithm with a sampling algorithm as shown in Algorithm 4, where we select the next token for each sample by the

probability distribution of subwords in the target vocabulary. We call this pure sampling because we did not add sampling temperature, top-k or top-p filtering. We perform the merging post-processing the same as the inner beam search.

Algorithm 4: InnerSampling

Data: Sample times K, max length T, $toks$

Result: Next word list

Initialization:

$s_0 \leftarrow \{(0, toks)\}$;

$Q \leftarrow \emptyset$;

for $i \leftarrow 1$ **to** K **do**

  $s \leftarrow s_0$;

  for $j \leftarrow 1$ **to** T **do**

    if s *reaches _ or* $< eos >$ **then**

      $Q.append(s)$;

      break;

    Sample s' from $Decoder(s[1])$;

    $s = s'$;

Sort Q by scores in descending order.;

$W = \{\}$;

foreach $s \in Q$ **do**

  $score, toks = s'$;

  $word = detokenize(toks)$;

  if $word \notin W$ **then**

    $W[word] = (score, toks)$

  else

    $W[word][0] += score$

return $list(W.items())$

Bibliography

[1] Akiko Aizawa, Frederic Bergeron, Junjie Chen, Fei Cheng, Katsuhiko Hayashi, Kentaro Inui, Hiroyoshi Ito, Daisuke Kawahara, Masaru Kitsuregawa, Hirokazu Kiyomaru, Masaki Kobayashi, Takashi Kodama, Sadao Kurohashi, Qianying Liu, Masaki Matsubara, Yusuke Miyao, Atsuyuki Morishima, Yugo Murawaki, Kazumasa Omura, Haiyue Song, Eiichiro Sumita, Shinji Suzuki, Ribeka Tanaka, Yu Tanaka, Masashi Toyoda, Nobuhiro Ueda, Honai Ueoka, Masao Utiyama, and Ying Zhong. A system for worldwide COVID-19 information aggregation. In Karin Verspoor, Kevin Bretonnel Cohen, Michael Conway, Berry de Bruijn, Mark Dredze, Rada Mihalcea, and Byron Wallace, editors, Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020, Online, December 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlpcovid19-2.13. URL `https://aclanthology.org/2020.nlpcovid19-2.13`.

[2] Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. Automatic text scoring using neural networks. 2016. doi: 10.18653/v1/P16-1068.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

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

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural ma-

chine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

[6] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://aclanthology.org/W05-0909.

[7] Khuyagbaatar Batsuren, Gábor Bella, and Fausto Giunchiglia. MorphyNet: a large multilingual database of derivational and inflectional morphology. In Garrett Nicolai, Kyle Gorman, and Ryan Cotterell, editors, Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 39–48, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.sigmorphon-1.5. URL https://aclanthology.org/2021.sigmorphon-1.5.

[8] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311, 1993. URL https://aclanthology.org/J93-2003.

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[10] Jacob Buckman and Graham Neubig. Neural lattice language models. Transactions of the Association for Computational Linguistics, 6:529–541,

2018. doi: 10.1162/tacl_a_00036. URL https://aclanthology.org/Q18-1036.

[11] Kris Cao and Laura Rimell. You should evaluate your language model on marginal likelihood over tokenisations, 2021.

[12] Kris Cao and Laura Rimell. You should evaluate your language model on marginal likelihood over tokenisations. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 2104–2114, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.161. URL https://aclanthology.org/2021.emnlp-main.161.

[13] Abhisek Chakrabarty, Raj Dabre, Chenchen Ding, Masao Utiyama, and Eiichiro Sumita. Improving low-resource NMT through relevance based linguistic features incorporation. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, Proceedings of the 28th International Conference on Computational Linguistics, pages 4263–4274, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.376. URL https://aclanthology.org/2020.coling-main.376.

[14] Abhisek Chakrabarty, Raj Dabre, Chenchen Ding, Masao Utiyama, and Eiichiro Sumita. Low-resource multilingual neural translation using linguistic feature-based relevance mechanisms. ACM Trans. Asian Low-Resour. Lang. Inf. Process., 22(7), jul 2023. ISSN 2375-4699. doi: 10.1145/3594631. URL https://doi.org/10.1145/3594631.

[15] William Chan, Yu Zhang, Quoc Le, and Navdeep Jaitly. Latent sequence decompositions, 2016.

[16] Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. Revisiting character-based neural machine translation with capacity and compression. In Proceedings of the 2018 Conference on Empirical

Methods in Natural Language Processing, pages 4295–4305, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1461. URL https://www.aclweb.org/anthology/D18-1461.

[17] Nadezhda Chirkova, Germán Kruszewski, Jos Rozen, and Marc Dymetman. Should you marginalize over possible tokenizations? In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 1–12, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.1. URL https://aclanthology.org/2023.acl-short.1.

[18] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

[19] Eldan Cohen and Christopher Beck. Empirical analysis of beam search performance degradation in neural sequence models. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 1290–1299. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/cohen19a.html.

[20] Marta R. Costa-jussà and José A. R. Fonollosa. Character-based neural machine translation. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 357–361, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2058. URL https://aclanthology.org/P16-2058.

[21] Raj Dabre and Atsushi Fujita. Combining sequence distillation and transfer learning for efficient low-resource neural machine translation models. In Proceedings of the Fifth Conference on Machine Translation, pages 492–502, Online, November 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.61.

[22] Raj Dabre, Atsushi Fujita, and Chenhui Chu. Exploiting multilingualism through multistage fine-tuning for low-resource neural machine translation. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 1410–1416, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1146. URL https://aclanthology.org/D19-1146.

[23] Hiroyuki Deguchi, Masao Utiyama, Akihiro Tamura, Takashi Ninomiya, and Eiichiro Sumita. Bilingual subword segmentation for neural machine translation. In Proceedings of the 28th International Conference on Computational Linguistics, pages 4287–4297, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.378. URL https://aclanthology.org/2020.coling-main.378.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[25] Chenchen Ding, Masao Utiyama, and Eiichiro Sumita. Similar Southeast Asian languages: Corpus-based case study on Thai-Laotian and Malay-Indonesian. In Proceedings of the 3rd Workshop on Asian Translation (WAT2016), pages 149–156, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://aclanthology.org/W16-4614.

[26] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual

representation learning by context prediction. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), December 2015.

[27] Chenhe Dong, Yinghui Li, Haifan Gong, Miaoxin Chen, Junxin Li, Ying Shen, and Min Yang. A survey of natural language generation. ACM Comput. Surv., 55(8), dec 2022. ISSN 0360-0300. doi: 10.1145/3554727. URL https://doi.org/10.1145/3554727.

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

[29] Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In Proceedings of the 28th International Conference on Computational Linguistics, pages 6903–6915, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.609. URL https://aclanthology.org/2020.coling-main.609.

[30] Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. Beyond english-centric multilingual machine translation, 2020.

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

[32] Matthias Gallé. Investigating the effectiveness of BPE: The power of shorter sequences. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 1375–1381, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1141. URL https://aclanthology.org/D19-1141.

[33] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, page 1243–1252. JMLR.org, 2017.

[34] Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. Is machine translation getting better over time? In Shuly Wintner, Sharon Goldwater, and Stefan Riezler, editors, Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 443–451, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi: 10.3115/v1/E14-1047. URL https://aclanthology.org/E14-1047.

[35] Edouard Grave, Sainbayar Sukhbaatar, Piotr Bojanowski, and Armand Joulin. Training hybrid language models by marginalizing over segmentations. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1477–1482, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1143. URL https://aclanthology.org/P19-1143.

[36] Alex Graves. Generating sequences with recurrent neural networks, 2013.

[37] Weiqi Gu, Haiyue Song, Chenhui Chu, and Sadao Kurohashi. Video-guided machine translation with spatial hierarchical attention network. In Jad Kabbara, Haitao Lin, Amandalynne Paullada, and Jannis Vamvas, editors, Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop, pages 87–92, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-srw.9. URL https://aclanthology.org/2021.acl-srw.9.

[38] Weidong Guo, Mingjun Zhao, Lusheng Zhang, Di Niu, Jinwen Luo, Zhenhua Liu, Zhenyang Li, and Jianbo Tang. LICHEE: Improving language model pre-training with multi-grained tokenization. In Findings of the

Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1383–1392, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.119. URL https://aclanthology.org/2021.findings-acl.119.

[39] Rohit Gupta, Laurent Besacier, Marc Dymetman, and Matthias Gallé. Character-based nmt with transformer, 2019.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[41] Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. Dynamic programming encoding for subword segmentation in neural machine translation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3042–3051, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.275. URL https://www.aclweb.org/anthology/2020.acl-main.275.

[42] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006. ISSN 0036-8075. doi: 10.1126/science.1127647. URL https://science.sciencemag.org/content/313/5786/504.

[43] Tatsuya Hiraoka. MaxMatch-dropout: Subword regularization for WordPiece. In Proceedings of the 29th International Conference on Computational Linguistics, pages 4864–4872, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.430.

[44] Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Keyaki Atsushi, and Naoaki Okazaki. テキストベクトルの重みづけを用いた タスクに対する単語分割の最適化. 自然言語処理, 28(2):479–507, 2021. doi: 10.5715/jnlp.28.479.

[45] Tatsuya Hiraoka, Sho Takase, Kei Uchiumi, Keyaki Atsushi, and Naoaki

Okazaki. 単語分割と後段モデルの損失値を用いた同時最適化. 自然言語処理, 29(1):112–143, 2022. doi: 10.5715/jnlp.29.112.

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

[47] Matthias Huck, Simon Riess, and Alexander Fraser. Target-side word segmentation strategies for neural machine translation. In Proceedings of the Second Conference on Machine Translation, pages 56–67, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4706. URL https://aclanthology.org/W17-4706.

[48] Aizhan Imankulova, Raj Dabre, Atsushi Fujita, and Kenji Imamura. Exploiting out-of-domain parallel data through multilingual transfer learning for low-resource neural machine translation. In Mikel Forcada, Andy Way, Barry Haddow, and Rico Sennrich, editors, Proceedings of Machine Translation Summit XVII: Research Track, pages 128–139, Dublin, Ireland, August 2019. European Association for Machine Translation. URL https://aclanthology.org/W19-6613.

[49] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1–10, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1001. URL https://aclanthology.org/P15-1001.

[50] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1700–1709, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1176.

[51] Nishant Kambhatla, Logan Born, and Anoop Sarkar. Auxiliary subword segmentations as related languages for low resource multilingual translation. In

Proceedings of the 23rd Annual Conference of the European Association for Machine Translation, pages 131–140, Ghent, Belgium, June 2022. European Association for Machine Translation. URL https://aclanthology.org/2022.eamt-1.16.

[52] Kazuya Kawakami, Chris Dyer, and Phil Blunsom. Learning to discover, ground and use words with segmental neural language models. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6429–6441, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1645. URL https://aclanthology.org/P19-1645.

[53] Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. Character-aware neural language models. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1), Mar. 2016. URL https://ojs.aaai.org/index.php/AAAI/article/view/10362.

[54] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[55] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. arXiv e-prints, art. arXiv:1412.6980, December 2014.

[56] Philipp Koehn. Statistical significance tests for machine translation evaluation. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pages 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-3250.

[57] Philipp Koehn and Kevin Knight. Empirical methods for compound splitting. In 10th Conference of the European Chapter of the Association for Computational Linguistics, Budapest, Hungary, April 2003. Association for Computational Linguistics. URL https://aclanthology.org/E03-1076.

[58] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine

Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P07-2045.

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

[60] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates, 2018.

[61] Taku Kudo. 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, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1007. URL https://www.aclweb.org/anthology/P18-1007.

[62] Taku Kudo and John Richardson. 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, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://www.aclweb.org/anthology/D18-2012.

[63] Yuxuan Lai, Yijia Liu, Yansong Feng, Songfang Huang, and Dongyan Zhao. Lattice-BERT: Leveraging multi-granularity representations in Chinese pretrained language models. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1716–1731, Online, June 2021. Asso-

ciation for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.137. URL https://aclanthology.org/2021.naacl-main.137.

[64] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. arXiv e-prints, art. arXiv:1607.06450, July 2016.

[65] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

[66] Jiahuan Li, Yutong Shen, Shujian Huang, Xinyu Dai, and Jiajun Chen. When is char better than subword: A systematic study of segmentation algorithms for neural machine translation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 543–549, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.69. URL https://aclanthology.org/2021.acl-short.69.

[67] Xiaonan Li, Hang Yan, Xipeng Qiu, and Xuanjing Huang. FLAT: Chinese NER using flat-lattice transformer. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6836–6842, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.611. URL https://aclanthology.org/2020.acl-main.611.

[68] Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du,

Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. Few-shot learning with multilingual generative language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 9019–9052, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.616. URL https://aclanthology.org/2022.emnlp-main.616.

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

[70] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[71] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual Denoising Pre-training for Neural Machine Translation. Transactions of the Association for Computational Linguistics, 8:726–742, 11 2020. ISSN 2307-387X. doi: 10.1162/tacl_a_00343. URL https://doi.org/10.1162/tacl_a_00343.

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

[73] Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 11–19, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1002. URL https://aclanthology.org/P15-1002.

[74] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

[75] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-5010. URL https://www.aclweb.org/anthology/P14-5010.

[76] Zhuoyuan Mao and Tetsuji Nakagawa. LEALLA: Learning lightweight language-agnostic sentence embeddings with knowledge distillation. In Andreas Vlachos and Isabelle Augenstein, editors, Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 1886–1894, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.138. URL https://aclanthology.org/2023.eacl-main.138.

[77] Zhuoyuan Mao and Yen Yu. Tuning llms with contrastive alignment instructions for machine translation in unseen, low-resource languages, 2024.

[78] Zhuoyuan Mao, Fabien Cromieres, Raj Dabre, Haiyue Song, and Sadao Kurohashi. Jass: Japanese-specific sequence to sequence pre-training for neural machine translation, 2020.

[79] Zhuoyuan Mao, Raj Dabre, Fabien Cromieres, Haiyue Song, Ryota Nakao, and Sadao Kurohashi. ニューラル機械翻訳のための言語知識に基づくマルチタスク事前学習. In 言語処理学会 第 26 回年次大会, pages 1061–1064, 2020.

[80] Zhuoyuan Mao, Prakhar Gupta, Chenhui Chu, Martin Jaggi, and Sadao Kurohashi. Lightweight cross-lingual sentence representation learning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, Proceedings of the 59th Annual Meeting of the Association for

Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 2902–2913, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.226. URL https://aclanthology.org/2021.acl-long.226.

[81] Zhuoyuan Mao, Prakhar Gupta, Chenhui Chu, Martin Jaggi, and Sadao Kurohashi. Learning cross-lingual sentence representations for multilingual document classification with token-level reconstruction. In 言語処理学会第 27 回年次大会, pages 1049–1053, 2021.

[82] Zhuoyuan Mao, Chenhui Chu, Raj Dabre, Haiyue Song, Zhen Wan, and Sadao Kurohashi. When do contrastive word alignments improve many-to-many neural machine translation? In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, Findings of the Association for Computational Linguistics: NAACL 2022, pages 1766–1775, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.134. URL https://aclanthology.org/2022.findings-naacl.134.

[83] Zhuoyuan Mao, Chenhui Chu, and Sadao Kurohashi. Linguistically driven multi-task pre-training for low-resource neural machine translation. ACM Trans. Asian Low-Resour. Lang. Inf. Process., 21(4), jan 2022. ISSN 2375-4699. doi: 10.1145/3491065. URL https://doi.org/10.1145/3491065.

[84] Zhuoyuan Mao, Chenhui Chu, and Sadao Kurohashi. EMS: efficient and effective massively multilingual sentence representation learning. CoRR, abs/2205.15744, 2022. doi: 10.48550/arXiv.2205.15744. URL https://doi.org/10.48550/arXiv.2205.15744.

[85] Zhuoyuan Mao, Chenhui Chu, and Sadao Kurohashi. Efficiently learning multilingual sentence representation for cross-lingual sentence classification. In 言語処理学会 第 29 回年次大会, pages 2830–2835, 2023.

[86] Zhuoyuan Mao, Raj Dabre, Qianying Liu, Haiyue Song, Chenhui Chu, and Sadao Kurohashi. Exploring the impact of layer normalization for zero-

shot neural machine translation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 1300–1316, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.112. URL https://aclanthology.org/2023.acl-short.112.

[87] Zhuoyuan Mao, Haiyue Song, Raj Dabre, Chenhui Chu, and Sadao Kurohashi. Variable-length neural interlingua representations for zero-shot neural machine translation. arXiv preprint arXiv:2305.10190, 2023.

[88] Shigeki Matsuda, Xinhui Hu, Yoshinori Shiga, Hideki Kashioka, Chiori Hori, Keiji Yasuda, Hideo Okuma, Masao Uchiyama, Eiichiro Sumita, Hisashi Kawai, and Satoshi Nakamura. Multilingual speech-to-speech translation system: Voicetra. In 2013 IEEE 14th International Conference on Mobile Data Management, volume 2, pages 229–233, 2013. doi: 10.1109/MDM.2013.99.

[89] Francois Meyer and Jan Buys. Subword segmental machine translation: Unifying segmentation and target sentence generation. In Findings of the Association for Computational Linguistics: ACL 2023, pages 2795–2809, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.175. URL https://aclanthology.org/2023.findings-acl.175.

[90] Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp, 2021.

[91] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[92] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification, 2016.

[93] Makoto Morishita, Jun Suzuki, and Masaaki Nagata. Improving neural machine translation by incorporating hierarchical subword features. In Proceedings of the 27th International Conference on Computational Linguistics, pages 618–629, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1052.

[94] Yasmin Moslem, Rejwanul Haque, John D. Kelleher, and Andy Way. Adaptive machine translation with large language models, 2023.

[95] JA Moyne. Georgetown automatic translation. Georgetown University, Washington, DC, USA, 1962.

[96] M. Nagao, J. Tsujii, K. Mitamura, H. Hirakawa, and M. Kume. A machine translation system from Japanese into English - another perspective of MT systems -. In COLING 1980 Volume 1: The 8th International Conference on Computational Linguistics, 1980. URL https://aclanthology.org/C80-1063.

[97] Makoto Nagao, Jun-ichi Tsujii, Koji Yada, and Toshihiro Kakimoto. An English Japanese machine translation system of the titles of scientific and engineering papers. In Coling 1982: Proceedings of the Ninth International Conference on Computational Linguistics, 1982. URL https://aclanthology.org/C82-1039.

[98] Graham Neubig. The Kyoto free translation task. http://www.phontron.com/kftt, 2011.

[99] OpenAI. Gpt-4 technical report, 2023.

[100] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pages 48–53, Minneapolis, Minnesota, June 2019. Asso-

ciation for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://www.aclweb.org/anthology/N19-4009.

[101] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

[102] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023.

[103] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. 2016.

[104] Maja Popović. chrF++: words helping character n-grams. In Proceedings of the Second Conference on Machine Translation, pages 612–618, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4770. URL https://aclanthology.org/W17-4770.

[105] Matt Post. A call for clarity in reporting BLEU scores. In Proceedings of the Third Conference on Machine Translation: Research Papers, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

[106] Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. BPE-dropout: Simple and effective subword regularization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1882–1892, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.170. URL https://www.aclweb.org/anthology/2020.acl-main.170.

[107] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.

[108] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019.

[109] Hammam Riza, Michael Purwoadi, Gunarso, Teduh Uliniansyah, Aw Ai Ti, Sharifah Mahani Aljunied, Luong Chi Mai, Vu Tat Thang, Nguyen Phuong Thai, Vichet Chea, Rapid Sun, Sethserey Sam, Sopheap Seng, Khin Mar Soe, Khin Thandar Nwet, Masao Utiyama, and Chenchen Ding. Introduction of the asian language treebank. In 2016 Conference of The Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA), pages 1–6, 2016. doi: 10.1109/ICSDA.2016.7918974.

[110] Raphael Rubino, Benjamin Marie, Raj Dabre, Atsushi Fujita, Masao Utiyama, and Eiichiro Sumita. Extremely low-resource neural machine translation for asian languages. Mach. Transl., 34(4):347–382, 2020. doi: 10.1007/s10590-020-09258-6. URL `https://doi.org/10.1007/s10590-020-09258-6`.

[111] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023.

[112] M. Schuster and K. Nakajima. Japanese and korean voice search. In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5149–5152, 2012. URL `https://ieeexplore.ieee.org/abstract/document/6289079`.

[113] Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT: Learning robust metrics for text generation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7881–7892, Online, July 2020. Association for Computational Linguistics. doi:

10.18653/v1/2020.acl-main.704. URL https://aclanthology.org/2020.acl-main.704.

[114] Rico Sennrich and Biao Zhang. Revisiting low-resource neural machine translation: A case study. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 211–221, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1021. URL https://aclanthology.org/P19-1021.

[115] Rico Sennrich, Barry Haddow, and Alexandra Birch. Edinburgh neural machine translation systems for WMT 16. In Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers, pages 371–376, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2323. URL https://aclanthology.org/W16-2323.

[116] Rico Sennrich, Barry Haddow, and Alexandra Birch. 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, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

[117] Uri Shaham and Omer Levy. Neural machine translation without embeddings. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 181–186, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.17. URL https://aclanthology.org/2021.naacl-main.17.

[118] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 464–468,

New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074. URL https://aclanthology.org/N18-2074.

[119] Haiyue Song, Raj Dabre, Atsushi Fujita, and Sadao Kurohashi. Coursera corpus mining and multistage fine-tuning for improving lectures translation. In Proceedings of the Twelfth Language Resources and Evaluation Conference, pages 3640–3649, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.449.

[120] Haiyue Song, Raj Dabre, Zhuoyuan Mao, Fei Cheng, Sadao Kurohashi, and Eiichiro Sumita. Pre-training via leveraging assisting languages for neural machine translation. In Shruti Rijhwani, Jiangming Liu, Yizhong Wang, and Rotem Dror, editors, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pages 279–285, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-srw.37. URL https://aclanthology.org/2020.acl-srw.37.

[121] Haiyue Song, Raj Dabre, Zhuoyuan Mao, Chenhui Chu, and Sadao Kurohashi. BERTSeg: BERT based unsupervised subword segmentation for neural machine translation. In Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 85–94, Online only, November 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.aacl-short.12.

[122] Haiyue Song, Raj Dabre, Chenhui Chu, Sadao Kurohashi, and Eiichiro Sumita. Selfseg: A self-supervised sub-word segmentation method for neural machine translation. 22(8), 2023. ISSN 2375-4699. doi: 10.1145/3610611. URL https://doi.org/10.1145/3610611.

[123] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MASS:

Masked Sequence to Sequence Pre-training for Language Generation. arXiv e-prints, art. arXiv:1905.02450, May 2019.

[124] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The Challenges of Clustering High Dimensional Data, pages 273–309. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-662-08968-2. doi: 10.1007/978-3-662-08968-2_16. URL https://doi.org/10.1007/978-3-662-08968-2_16.

[125] Zhiqing Sun and Zhi-Hong Deng. Unsupervised neural word segmentation for Chinese via segmental language modeling. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4915–4920, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1531. URL https://aclanthology.org/D18-1531.

[126] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[127] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[128] Sho Takase, Tatsuya Hiraoka, and Naoaki Okazaki. Single model ensemble for subword regularized models in low-resource machine translation. In Findings of the Association for Computational Linguistics: ACL 2022, pages 2536–2541, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.199. URL https://aclanthology.org/2022.findings-acl.199.

[129] Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Met-

zler. Charformer: Fast character transformers via gradient-based subword tokenization, 2021.

[130] Yee Whye Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pages 985–992, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220299. URL https://aclanthology.org/P06-1124.

[131] Ye Kyaw Thu, Win Pa Pa, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. Introducing the Asian language treebank (ALT). In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), pages 1574–1578, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L16-1249.

[132] Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. Juman++: A morphological analysis toolkit for scriptio continua. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 54–59, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2010. URL https://www.aclweb.org/anthology/D18-2010.

[133] Peter Toma. Systran as a multilingual machine translation system. In Proceedings of the Third European Congress on Information Systems and Networks, overcoming the language barrier, pages 569–581, 1977.

[134] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev,

Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[135] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[136] Jesse Vig. A multiscale visualization of attention in the transformer model. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 37–42, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3007. URL https://www.aclweb.org/anthology/P19-3007.

[137] David Vilar, Markus Freitag, Colin Cherry, Jiaming Luo, Viresh Ratnakar, and George Foster. Prompting palm for translation: Assessing strategies and performance, 2022.

[138] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning, ICML '08, page 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390294. URL https://doi.org/10.1145/1390156.1390294.

[139] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[140] Zhen Wan, Fei Cheng, Qianying Liu, Zhuoyuan Mao, Haiyue Song, and Sadao Kurohashi. Relation extraction with weighted contrastive pre-training on distant supervision. In Andreas Vlachos and Isabelle Augenstein, editors, Findings of the Association for Computational Linguistics: EACL 2023, pages 2580–2585, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.195. URL https://aclanthology.org/2023.findings-eacl.195.

[141] Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. GPT-RE: In-context learning for relation extraction using large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 3534–3547, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.214. URL https://aclanthology.org/2023.emnlp-main.214.

[142] Changhan Wang, Kyunghyun Cho, and Jiatao Gu. Neural machine translation with byte-level subwords, 2019.

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

[144] Yining Wang, Long Zhou, Jiajun Zhang, and Chengqing Zong. Word, subword or character? an empirical study of granularity in chinese-english nmt, 2017.

[145] Donglai Wei, Joseph J. Lim, Andrew Zisserman, and William T. Freeman. Learning and using the arrow of time. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[146] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022.

[147] 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, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[148] Fengshun Xiao, Jiangtong Li, Hai Zhao, Rui Wang, and Kehai Chen. Lattice-based transformer encoder for neural machine translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3090–3097, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1298. URL https://aclanthology.org/P19-1298.

[149] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. 2020.

[150] Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. Vocabulary learning via optimal transport for neural machine translation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 7361–7373, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.571. URL https://aclanthology.org/2021.acl-long.571.

[151] Biao Zhang, Barry Haddow, and Alexandra Birch. Prompting large language model for machine translation: A case study, 2023.

[152] Pei Zhang, Niyu Ge, Boxing Chen, and Kai Fan. Lattice transformer for speech translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6475–6484, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1649. URL https://aclanthology.org/P19-1649.

[153] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In European conference on computer vision, pages 649–666. Springer, 2016.

[154] Giulio Zhou. Morphological Zero-Shot Neural Machine Translation. University of Edinburgh, 2018.

List of Major Publications

[1] Haiyue Song, Raj Dabre, Chenhui Chu, Sadao Kurohashi, and Eiichiro Sumita. SelfSeg: A Self-supervised Sub-word Segmentation Method for Neural Machine Translation. In ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), Volume 22, Issue 8, pages 1–24, doi 10.1145/3610611, 2023.

[2] Haiyue Song, Raj Dabre, Zhuoyuan Mao, Chenhui Chu, and Sadao Kurohashi. BERTSeg: BERT Based Unsupervised Subword Segmentation for Neural Machine Translation. In Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (AACL-IJCNLP2022), pages 85-94, 2022.

[3] Haiyue Song, Zhuoyuan Mao, Raj Dabre, Chenhui Chu, and Sadao Kurohashi. DiverSeg: Leveraging Diverse Segmentations with Cross-granularity Alignment for Neural Machine Translation. In Journal of Natural Language Processing (JNLP), to appear in Vol.31 No.1, Mar. 2024.

[4] Haiyue Song, Francois Meyer, Raj Dabre, Chenhui Chu, and Sadao Kurohashi. SubMerge: Merging Equivalent Subword Tokenizations for Subword Regularized Models in Neural Machine Translation. Submitted to The 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), 2024. **under review**

[5] Haiyue Song, Raj Dabre, Chenhui Chu, Atsushi Fujita, and Sadao Kurohashi. Bilingual Corpus Mining and Multistage Fine-Tuning for Improving Machine Translation of Lecture Transcripts. Submitted to Journal of Information Processing. **under review**

[6] Akiko Aizawa, Frederic Bergeron, Junjie Chen, Fei Cheng, Katsuhiko Hayashi, Kentaro Inui, Hiroyoshi Ito, Daisuke Kawahara, Masaru Kitsuregawa, Hirokazu Kiyomaru, Masaki Kobayashi, Takashi Kodama, Sadao Kurohashi, Qianying Liu, Masaki Matsubara, Yusuke Miyao, Atsuyuki Morishima,

Yugo Murawaki, Kazumasa Omura, Haiyue Song, Eiichiro Sumita, Shinji Suzuki, Ribeka Tanaka, Yu Tanaka, Masashi Toyoda, Nobuhiro Ueda, Honai Ueoka, Masao Utiyama, and Ying Zhong. A System for Worldwide COVID-19 Information Aggregation. In Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020, 2020.

[7] Haiyue Song, Raj Dabre, Zhuoyuan Mao, Fei Cheng, Sadao Kurohashi, and Eiichiro Sumita. Pre-training via Leveraging Assisting Languages for Neural Machine Translation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop (ACL-SRW2020), pages 279–285, 2020.

[8] Haiyue Song, Raj Dabre, Atsushi Fujita, and Sadao Kurohashi. Coursera Corpus Mining and Multistage Fine-Tuning for Improving Lectures Translation. In Proceedings of the Twelfth Language Resources and Evaluation Conference (LREC2020), pages 3640—3649, 2020.

List of Other Publications

[9] Haiyue Song, Raj Dabre, Chenhui Chu, and Sadao Kurohashi. Large Pre-trained Language Models with Multilingual Prompt for Japanese Natural Language Tasks. In Proceedings of the 29th Annual Meeting of the Association for Natural Language Processing (NLP2023), pages 810–814, 2023.

[10] Haiyue Song, Raj Dabre, Zhuoyuan Mao, Chenhui Chu, and Sadao Kurohashi. Representative Data Selection for Sequence-to-Sequence Pre-training. In Proceedings of the 28th Annual Meeting of the Association for Natural Language Processing (NLP2022), pages 1–5, 2022.

[11] Haiyue Song, Raj Dabre, Chenhui Chu, Sadao Kurohashi, and Eiichiro Sumita. Self-supervised Dynamic Programming Encoding for Neural Machine Translation. In Proceedings of the 27th Annual Meeting of the Association for Natural Language Processing (NLP2021), pages 632–636, 2021.

[12] Haiyue Song, Raj Dabre, Atsushi Fujita, and Sadao Kurohashi. Domain Adaptation of Neural Machine Translation through Multistage Fine-Tuning. In Proceedings of the 26th Annual Meeting of the Association for Natural Language Processing (NLP2020), pages 461–464, 2020.

[13] Haiyue Song, Chengwen Xu, Qiang Xu, Zhuoran Song, Naifeng Jing, Xiaoyao Liang, and Li Jiang. Invocation-driven neural approximate computing with a multiclass-classifier and multiple approximators. In In Proceedings of the International Conference on Computer-Aided Design (ICCAD2019), pages 1–8, 2019.

[14] Haiyue Song, Xiang Song, Tianjian Li, Hao Dong, Naifeng Jing, Xiaoyao Liang, and Li Jiang. A FPGA Friendly Approximate Computing Framework with Hybrid Neural Networks: (Abstract Only). In In Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA2018), page 286, 2018.

[15] Li Jiang, Zhuoran Song, Haiyue Song, Chengwen Xu, Qiang Xu, Naifeng Jing, Weifeng Zhang, Xiaoyao Liang. Energy-Efficient and Quality-Assured Approximate Computing Framework Using a Co-Training Method. In ACM Transactions on Design Automation of Electronic Systems (TODAES), pages.59:1–59:25, 2019.

[16] Zhuoyuan Mao, Raj Dabre, Qianying Liu, Haiyue Song, Chenhui Chu, and Sadao Kurohashi. Exploring the Impact of Layer Normalization for Zero-shot Neural Machine Translation. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL2023), pages 1300–1316, 2023.

[17] Zhuoyuan Mao, Haiyue Song, Raj Dabre, Chenhui Chu and Sadao Kurohashi. Variable-length Neural Interlingua Representations for Zero-shot Neural Machine Translation. In the 1st Workshop on Multilingual, Multimodal and Multitask Language Generation, pages 16–25, 2023.

[18] Zhuoyuan Mao, Chenhui Chu, Raj Dabre, Haiyue Song, Zhen Wan and Sadao Kurohashi. When do Contrastive Word Alignments Improve Many-

to-many Neural Machine Translation? In Findings of the Association for Computational Linguistics: NAACL 2022, pages 1766–1775, 2022.

[19] Zhuoyuan Mao, Fabien Cromieres, Raj Dabre, Haiyue Song and Sadao Kurohashi. JASS: Japanese-specific Sequence to Sequence Pre-training for Neural Machine Translation. In Proceedings of the 12th Language Resources and Evaluation Conference, pages 3683–3691, 2020.

[20] Zhuoyuan Mao, Raj Dabre, Fabien Cromieres, Haiyue Song, Ryota Nakao and Sadao Kurohashi. ニューラル機械翻訳のための言語知識に基づくマルチタスク事前学習. In Proceedings of the 26th Annual Meeting of the Association for Natural Language Processing (NLP2020), pages 1061–1064, 2020. (in Japanese).

[21] Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, Sadao Kurohashi. GPT-RE: In-context Learning for Relation Extraction using Large Language Models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP2023), pages 3534–3547, 2023.

[22] Zhen Wan, Fei Cheng, Qianying Liu, Zhuoyuan Mao, Haiyue Song and Sadao Kurohashi. Relation Extraction with Weighted Contrastive Pre-training on Distant Supervision. In Findings of the Association for Computational Linguistics: EACL 2023, pages 2580–2585, 2023.

[23] Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Sadao Kurohashi. Improving Medical Relation Extraction with Distantly Supervised Pre-training. In Proceedings of the 28th Annual Meeting of the Association for Natural Language Processing (NLP2022), pages 610–614, 2022.

[24] Weiqi Gu, Haiyue Song, Chenhui Chu, Sadao Kurohashi. Spatial Hierarchical Attention Network Based Video-guided Machine Translation. In Journal of Information Processing (JIP), 31, pages 299–307, 2023.

[25] Weiqi Gu, Haiyue Song, Chenhui Chu, and Sadao Kurohashi. Video-guided Machine Translation with Spatial Hierarchical Attention Network. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop (ACL-SRW 2023), pages 87–92, 2021.

[26] Weiqi Gu, Haiyue Song, Chenhui Chu, and Sadao Kurohashi. Video-guided Machine Translation with Spatial Hierarchical Attention Network Encoder. In Proceedings of the 27th Annual Meeting of the Association for Natural Language Processing (NLP2021), 2021.