Sequence Generation with Mixed Representations

Lijun Wu* [1 2] **Shufang Xie*** [1] **Yingce Xia** [1] **Yang Fan** [3] **Tao Qin** [1] **Jianhuang Lai** [2] **Tie-Yan Liu** [1]

Abstract

Tokenization is the first step of many natural language processing (NLP) tasks and plays an important role for neural NLP models. Tokenization methods such as byte-pair encoding and SentencePiece, which can greatly reduce the large vocabulary size and deal with out-of-vocabulary words, have shown to be effective and are widely adopted for sequence generation tasks. While various tokenization methods exist, there is no common acknowledgement which one is the best. In this work, we propose to leverage the mixed representations from different tokenizers for sequence generation tasks, which can take the advantages of each individual tokenization method. Specifically, we introduce a new model architecture to incorporate mixed representations and a co-teaching algorithm to better utilize the diversity of different tokenization methods. Our approach achieves significant improvements on neural machine translation tasks with six language pairs, as well as an abstractive summarization task.

1. Introduction

Natural language processing (NLP) has achieved great success with deep neural networks in recent years (Deng & Liu, 2018; Zhang et al., 2015; Deng et al., 2013; Wu et al., 2016; Hassan et al., 2018). For neural based NLP models, tokenization, which chops raw sequence up into pieces, is the first step and plays the most important role in the text preprocessing. Previously, tokenization is always performed on word level (Arppe et al., 2005; Bahdanau et al., 2014), which splits a raw sentence by spaces and applies language-specific rules to punctuation marks, or character level (Kim et al., 2016; Lee et al., 2017), which directly segments words into individual characters. However, previous works revealed that such tokenization methods have their own drawbacks (Ling et al., 2015; Cherry et al., 2018; Luong & Manning, 2016) and often lead to inaccurate results.

Table 1. A German sentence example (Ori stands for original) of tokenization results by BPE, WordPiece (WP), SentencePiece (SP) tokenizers. Different subwords are highlighted in **bold** font. @@ and ▁ represent the boundaries of subwords.

	Results
Ori	und diese einfachen themen sind eigentlich keine komplexen wissenschaftlichen zusammenhänge
BPE	und diese einfachen **themen** sind eigentlich keine **komplex@@ en wissenschaftlichen zusammen@@ hän@@ ge**
WP	und▁ diese▁ einfachen▁ **themen**▁ sind▁ eigentlich▁ keine▁ **komplexe n▁ wissenschaftlichen▁ zusammen hä nge▁**
SP	▁und ▁diese ▁einfachen **▁them en** ▁sind ▁eigentlich ▁keine **▁komplexen ▁wissenschaft lichen ▁zusammenhänge**

Recently, there are several types of tokenization methods shown to be very effective, which split sentences into subword units according to the statistics of consequent characters. Byte-pair encoding (BPE) (Sennrich et al., 2015) constructs the vocabulary based on the subword frequency, and word level tokenization should be applied first before using BPE. SentencePiece (SP)[1] (Kudo, 2018; Kudo & Richardson, 2018) and WordPiece (WP) (Schuster & Nakajima, 2012) tokenizers leverage language models to build the vocabulary, which can be applied to the raw text data without word tokenization. Table 1 shows the results of three tokenizers for one example of German sentence and they differ a lot with each other. For example, BPE segments the word "komplexen" into "komplex" and "en", while WP segments it into "komplexe" and "n" but SP keeps it as is. Despite the effectiveness of various tokenizers, there is no common acknowledgement about which approach is the best across different tasks. Taking machine translation as an example, preliminary experiments (see Table 2) show that no specific tokenizer can universally perform best across different datasets. Considering each tokenization method has its own strengths and unique characteristics, instead of inventing an-

*Equal contribution [1]Microsoft Research, Beijing, China [2]Sun Yat-sen University, Guangzhou, China [3]University of Science and Technology of China, Hefei, China. Correspondence to: Lijun Wu <Lijun.Wu@microsoft.com>, Yingce Xia <Yingce.Xia@microsoft.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

[1]To be concrete, we refer the unigram language model in (Kudo, 2018) as SentencePiece.

Table 2. Performance of BPE, WordPiece (WP), SentencePiece (SP) tokenizers on different IWSLT translation tasks. Details of experiment settings are left in Section 4.1.

Dataset	BPE	SP	WP
German→English	34.84	34.77	**34.91**
English→German	**28.80**	28.45	28.71
English→Romanian	24.56	**24.67**	24.63

other tokenizer, can we leverage these different tokenization methods to build better sequence learning models?

In this work, we leverage multiple tokenizers in a single model. We focus on the sequence generation tasks in this paper, whose backbone is the encoder-decoder network. We propose a new model architecture and a co-teaching algorithm: (1) The new architecture incorporates different tokenizers into a unified model to get the mixed representations. Specifically, two encoders are leveraged to deal with the different tokenized sequences as inputs. Each layer in one encoder interacts with the previous layer of the other encoder to generate the mixed representations. Correspondingly, there are two decoders to generate sequences in two tokenized ways, both of which take the mixed representations from the encoders as inputs. In this way, the model can benefit from the enriched representations by combining different tokenizers. (2) Besides the model structure, we propose a co-teaching algorithm to further utilize the diversity of outputs from different tokenizers. In the decoder side, our model is able to generate sentences in different tokenization ways. Thus, we let each decoder learn from the other one with different tokenizers rounds by rounds iteratively. Specially, our approach works well in the low-resource setting since we have mixed representations from different tokenizers instead of only one, which greatly increases the data diversity. This is because the frequency based methods and language model based methods have their own limitation in the face of paucity of data.

We evaluate our model and training algorithm in two standard sequence generation tasks, machine translation and abstractive summarization. On six translation language pairs and totally 12 translation tasks, such as English↔German, English↔Dutch and English↔Romanian, our approach outperforms baselines by more than 1.0 BLEU points. For abstractive summarization, experimental results also show consistent improvements compared with only one tokenizer utilized. Our code is provided in `https://github.com/apeterswu/fairseq_mix`.

2. Related Work

In this section, we first introduce the background of several tokenization approaches. Then we introduce some recent works of leveraging different tokenizers.

2.1. Tokenization Approaches

We describe the details of different tokenization approaches here. BPE (Sennrich et al., 2015) tokenizer initializes the vocabulary with all the characters and builds the final vocabulary by iteratively merging frequent n-gram characters. Similarly, WordPiece (Schuster & Nakajima, 2012) also constructs the vocabulary from characters. Different from BPE, WordPiece forms a new subword according to the n-gram likelihood on the training data instead of the next highest frequency pair. SentencePiece (Kudo, 2018; Kudo & Richardson, 2018) is based on the assumption that all subword occurrences are independent and a tokenized sequence is produced by the product of subword occurrence probabilities. Therefore, SentencePiece selects and builds the subword dictionary on the word occurrence and the loss of each subword. Both WordPiece and SentencePiece leverage language models to build their vocabularies.

2.2. Leveraging Tokenization Approaches

Besides inventing new tokenizers to handle rare and unseen words for text-based sequence learning, there are several pieces of works focusing on how to leverage existed tokenizers from different aspects recently. Srinivasan et al. (2019) propose to use BPE with multiple merge operations to build the vocabulary in the target side of a multitask learning model. Pan et al. (2020) introduce a new morphological subword tokenization method by using morpheme segmentation and BPE together for morphological rich language, but only in the source side. Moreover, Wang et al. (2019b) focus on the multilingual machine translation setting. They design a multilingual lexicon encoding framework named soft decoupled encoding, which is designed to present a word by its spelling through character and its semantic meaning by a shared latent embedding. Provilkov et al. (2019) stochastically corrupt the segmentation procedure of BPE and produce multiple segmentations in BPE framework. Our work differs from above works in that we leverage existed different subword tokenization methods effectively on both source and target languages. Also, we propose a co-teaching algorithm to better leverage different tokenization methods.

3. Architecture and Co-Teaching Algorithm

We first introduce our proposed architecture in Section 3.1 and the co-teaching algorithm in Section 3.2. In this work, we mainly introduce mixed representations obtained from BPE and SP tokenizers. It is easy to extend our algorithm to any other two types of tokenizers.

3.1. Architecture

Our model is based on Transformer (Vaswani et al., 2017). The overall architecture for our model is shown in Figure 1.

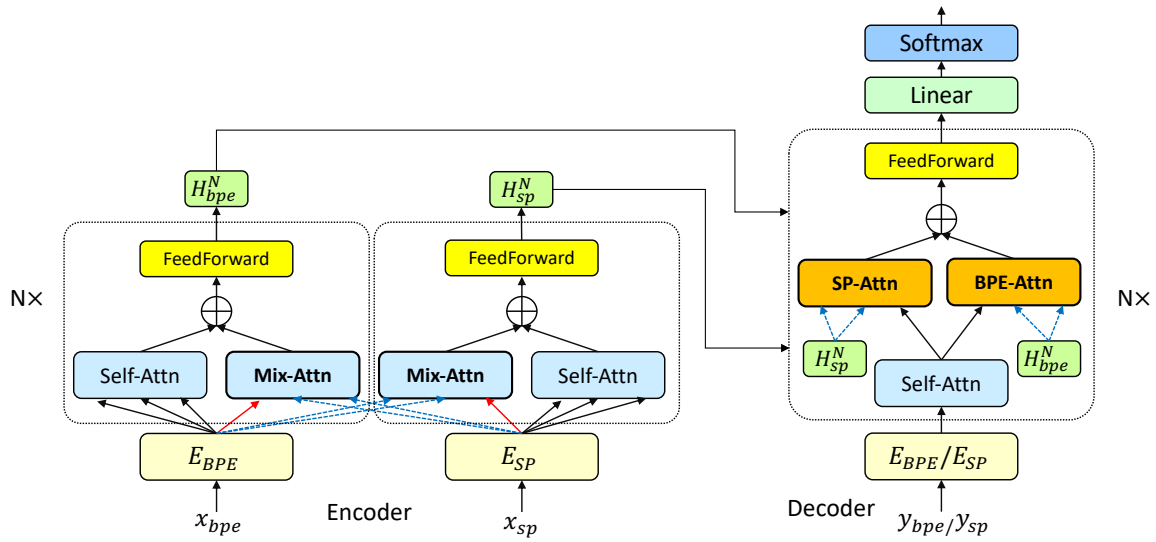

Figure 1. The overall framework for our proposed model. The important components mix-attention (Mix-Attn in figure), SP-enc-attention (SP-Attn in figure) and BPE-enc-attention (BPE-Attn in figure) are in **bold** font. For mix-attention, the input of query (Q) are red lines, while the blue dash lines stand for the key (K) and value (V) used in attention computation. For SP-enc-attention and BPE-enc-attention, the K and V are also in blue dash lines. Here we only show one decoder since the other one has same components except the different tokenized inputs. Note that the residual connections and the layer norm operation after each component are same as the standard Transformer, we omit them here to save space.

Different from Transformer, there are two N-layer encoders and two N-layer decoders in our model. The two encoders are mixed through attention layers.

Encoder For each input sentence x, we first process it by two tokenizers, BPE and SP, and then obtain x_{bpe} and x_{sp}, respectively. There are two embeddings in our model, one for BPE sequences (denoted as E_{bpe}) and one for SP sequences (denoted as E_{sp}). We leverage two encoders, one starting with processing E_{bpe} (denoted as $\texttt{enc}_{bpe}$) and the other with E_{sp} (denoted as $\texttt{enc}_{sp}$). Two types of attention layers are used in each encoder, one is the self-attention layer, which also exists in the standard Transformer (Vaswani et al., 2017), and the other one is the mix-attention layer we proposed, which is used to fuse the BPE representations and SP representations. They are shown as "Self-Attn" and "Mix-Attn" in the Figure 1, respectively.

Mathematically, let H^l_{bpe} and H^l_{sp} be the representations in l-th layer of $\texttt{enc}_{bpe}$ and $\texttt{enc}_{sp}$ respectively, $l \in [N]$. Specially, let H^0_{bpe} and H^0_{sp} denote the embedding representations of x_{bpe} and x_{sp}. For any $l \in [N]$, H^l_{bpe} and H^l_{sp} are obtained as follows:

$$H^l_{bpe} = \texttt{attn}^{bpe}_s(H^{l-1}_{bpe}, H^{l-1}_{bpe}, H^{l-1}_{bpe}) + \texttt{attn}^{bpe}_m(H^{l-1}_{bpe}, H^{l-1}_{sp}, H^{l-1}_{sp}); \quad (1)$$

$$H^l_{sp} = \texttt{attn}^{sp}_s(H^{l-1}_{sp}, H^{l-1}_{sp}, H^{l-1}_{sp}) + \texttt{attn}^{sp}_m(H^{l-1}_{sp}, H^{l-1}_{bpe}, H^{l-1}_{bpe}); \quad (2)$$

where $\texttt{attn}^{bpe}_s$ and $\texttt{attn}^{bpe}_m$ represent the self-attention layer and mix-attention layer in $\texttt{enc}_{bpe}$, $\texttt{attn}^{sp}_s$ and $\texttt{attn}^{sp}_m$ represent the corresponding layers in $\texttt{enc}_{sp}$.

Each attention layer $\texttt{attn}(Q, K, V)$ is implemented as follows:

$$\texttt{softmax}\Big(\frac{(QW_Q)(KW_K)^\top}{\sqrt{d}}(VQ_V)\Big), \quad (3)$$

where Q, K, V denote query, key and value respectively, d is the embedding dimension, and W_Q, W_K, W_V are the parameters to be learned.

Compared with standard Transformer, in each encoder (and later, the decoder), we only revise the attention layers. For simplicity, we do not include the formulations for feed-forward layers, layer normalization (Ba et al., 2016) or residual connection of our model in the main text. Details can be found at our code in the code.

From Eqn.(1), we can see that H^l_{bpe} is the sum of two components, one is related to the output of previous layer only, and the other is related to the $(l-1)$'th layer of $\texttt{enc}_{sp}$. In such a way, for each subword obtained by BPE, the representation is enriched by that from SentencePiece. Similarly, subwords obtained by SentencePiece are enriched by BPE too. Therefore, H^l_{bpe} and H^l_{sp} are mixed representations for each individual subword, which are enhanced to provide more comprehensive contexts. The final representations for the encoders are H^N_{bpe} and H^N_{sp} outputted by the last layer.

Decoder The mixed representation outputted from the encoders is a benefit for the decoding process. There are two decoders in our architecture, $\texttt{dec}_{bpe}$ and $\texttt{dec}_{sp}$, which are used to generate sequences with BPE and SP tokenizations respectively. The way that we fuse the representations in the decoder side is to let the encoder-decoder attention layers attend to the output from both $\texttt{enc}_{bpe}$ and $\texttt{enc}_{sp}$. That is, for each decoder, we introduce another two attention layers: a BPE-enc-attention layer, which bridges the output of $\texttt{enc}_{bpe}$ to the decoder, and a SP-enc-attention layer, which bridges $\texttt{enc}_{sp}$ and decoder together.

Let S^l_{bpe} and S^l_{sp} be the representations in the l-th layer of $\texttt{dec}_{bpe}$ and $\texttt{dec}_{sp}$. S^0_{bpe} and S^0_{sp} are embeddings for sentences y_{bpe} and y_{sp}. The two decoders work as follows:

$$\tilde{S}^l_{bpe} = \texttt{attn}^{bpe}_s(S^{l-1}_{bpe}, S^{l-1}_{bpe}, S^{l-1}_{bpe}), \quad (4)$$

$$\tilde{S}^l_{sp} = \texttt{attn}^{sp}_s(S^{l-1}_{sp}, S^{l-1}_{sp}, S^{l-1}_{sp}), \quad (5)$$

$$S^l_{bpe} = \texttt{attn}^{bpe}_{bpe}(\tilde{S}^l_{bpe}, H^N_{bpe}, H^N_{bpe}) + \texttt{attn}^{bpe}_{sp}(\tilde{S}^l_{bpe}, H^N_{sp}, H^N_{sp}), \quad (6)$$

$$S^l_{sp} = \texttt{attn}^{sp}_{sp}(\tilde{S}^l_{sp}, H^N_{sp}, H^N_{sp}) + \texttt{attn}^{sp}_{bpe}(\tilde{S}^l_{sp}, H^N_{bpe}, H^N_{bpe}), \quad (7)$$

All $\texttt{attn}$'s in the above formulation are implemented as Eqn.(3) with different parameters, where the superscripts represent which decoder the attention layer belongs to, and the subscripts s, bpe and sp represent self-attention, attending to outputs of $\texttt{enc}_{bpe}$ and attending to outputs to $\texttt{enc}_{sp}$.

We can see that the two decoders first leverage self-attention layers to preprocess the output S^{l-1}_{bpe}, S^{l-1}_{sp} from previous layers, and obtain the results $\tilde{S}^l_{bpe}$, $\tilde{S}^l_{sp}$. Then, $\texttt{dec}_{bpe}$ and $\texttt{dec}_{sp}$ leverage $\tilde{S}^l_{bpe}$ and $\tilde{S}^l_{sp}$ as queries respectively to bridge the relation with both H^N_{bpe} and H^N_{sp}, which are encoder representations of the input sequence tokenized in different ways. In this way, S^l_{bpe} and S^l_{sp} can leverage both H^N_{bpe} outputted by the BPE encoder and H^N_{sp} outputted by the SP encoder, which can benefit the corresponding decoder to have enriched representations during decoding process. After the final layer outputs S^N_{bpe} and S^N_{sp} from $\texttt{dec}_{bpe}$ and $\texttt{dec}_{sp}$, they will be mapped via a linear transformation and then the $\texttt{softmax}$ operation. Based on the output probability, the BPE decoder generates the sentence tokenized in BPE format, and the SP decoder generates the output sentence tokenized in SP format.

In our proposed method, we mix up the two tokenized sequences, one with BPE and the other with SP, by summing their representations. In Section 5.1, we discuss an alternative way, stack representation, where the self-attention layer and the mix-attention layer are sequentially stacked instead of parallel adding. Experimental results show that adding up the representations is better. For decoders, actually we can also add mix-attention component as used in the encoder. However, this will make the beam search decoding quite complex when selecting the top-k beams so that we leave it as future work.

3.2. Co-Teaching Algorithm

The two encoders interact through the mix-attention layers to generate mix representations, while the two decoders work independently. To better utilize the diversity brought by different tokenizers, we further propose a co-teaching algorithm, by which the two decoders can teach each other.

Denote the original bilingual data as $D = \{(x^i, y^i)\}_{i=1}^M$, where M is the number of training sentence pairs. For simplicity, define $[M] = \{1, 2, \cdots, M\}$. After applying BPE and SP tokenizers to the data pairs in D, we obtain a dataset made up of four-element tuples: $D_{\text{mix}} = \{(x^i_{bpe}, y^i_{bpe}, x^i_{sp}, y^i_{sp})\}_{i=1}^M$, where x^i_{bpe}, y^i_{bpe} are the outputs of BPE tokenizer, and x^i_{sp}, y^i_{sp} are obtained by using SP tokenizer.

Let θ^e denote the parameter of the encoders (including both $\texttt{enc}_{bpe}$ and $\texttt{enc}_{sp}$), θ^d_{bpe} and θ^d_{sp} are the parameters of the decoder for BPE and that for SP respectively. Given any $(x^i_{bpe}, y^i_{bpe}, x^i_{sp}, y^i_{sp})$ in D_{mix}, let $P(y^i_{bpe}|x^i_{bpe}, x^i_{sp}; \theta^e, \theta^d_{bpe})$ denote the probability that y^i_{bpe} is obtained by the decoder $\texttt{dec}_{bpe}$, with mixed representation provided by x^i_{bpe} and x^i_{sp}. Similarly, $P(y^i_{sp}|x^i_{bpe}, x^i_{sp}; \theta^e, \theta^d_{sp})$ represents the probability that y^i_{sp} is obtained by the decoder $\texttt{dec}_{sp}$. No matter for computation of each probability, both encoders are involved.

In our proposed co-teaching algorithm, the loss function consists of two parts:

(Part 1) *Negative log-likelihood loss*: Following the common practice in sequence to sequence learning, we use negative log-likelihood loss for our model training. Since we have two decoders with different tokenizers, given D_{mix}, the loss function is designed as follows:

$$\ell_{\text{nll}} = -\frac{1}{M}\sum_{i=1}^{M} \log P(y^i_{bpe}|x^i_{bpe}, x^i_{sp}; \theta^e, \theta^d_{bpe}) - \frac{1}{M}\sum_{i=1}^{M} \log P(y^i_{sp}|x^i_{bpe}, x^i_{sp}; \theta^e, \theta^d_{sp}). \quad (8)$$

In Eqn.(8), we can see that no matter which decoder we use, the two encoders are both updated, since their mixed representations are used in each decoder.

(Part 2) *Co-teaching loss*: The co-teaching is implemented in a distillation way (Kim et al., 2016; Hinton et al., 2015). Let θ^e_{t-1}, $\theta^d_{bpe,t-1}$ and $\theta^d_{sp,t-1}$ denote the parameters obtained at epoch $t-1$, where t is a positive integer. With these parameters, we first generate some samples as follows:

(i) $\forall i \in [M]$, sample $\hat{y}^i_{bpe} \sim P(\cdot|x^i_{bpe}, x^i_{sp}; \theta^e_{t-1}, \theta^d_{bpe,t-1})$; detokenize $\hat{y}^i_{bpe}$ to the raw sentence and re-tokenize it with SP to get $\hat{y}^i_{bpe\to sp}$;

(ii) $\forall i \in [M]$, sample $\hat{y}^i_{sp} \sim P(\cdot|x^i_{bpe}, x^i_{sp}; \theta^e_{t-1}, \theta^d_{sp,t-1})$; detokenize $\hat{y}^i_{sp}$ to the raw sentence and re-tokenize it with BPE to get $\hat{y}^i_{sp\to bpe}$;

The co-teaching loss is defined upon the above data. Mathematically,

$$\begin{aligned}\ell_{\text{co}} = &-\frac{1}{M}\sum_{i=1}^{M}\log P(\hat{y}^i_{sp\to bpe}|x^i_{bpe}, x^i_{sp}; \theta^e, \theta^d_{bpe}) \\ &-\frac{1}{M}\sum_{i=1}^{M}\log P(\hat{y}^i_{bpe\to sp}|x^i_{bpe}, x^i_{sp}; \theta^e, \theta^d_{sp}).\end{aligned} \tag{9}$$

Co-teaching is conducted by generating a new sample for the other decoder to be taught. In this way, the two decoders interact with each other through data and the diversity from different tokenizers is leveraged.

The overall training loss is:

$$\ell_{\text{all}} = \ell_{\text{nll}} + \ell_{\text{co}}. \tag{10}$$

We can obtain $\theta^e_t, \theta^e_{bpe,t}, \theta^d_{sp,t} = \arg\min \ell_{\text{all}}$. Then we can go back to (part 2), re-sample the data using the latest model, and optimize Eqn.(10) until convergence. To stabilize the training, we can first optimize Eqn.(8) until convergence, and then introduce the co-teaching loss.

4. Experiments

To evaluate our proposed model and training algorithm, we conduct experiments on two standard sequence generation tasks: machine translation and abstractive summarization. The machine translation task aims to translate the sentences from source language to target language, while the goal for abstractive summarization is to summarize a long document with a few sentences.

4.1. Machine Translation

We introduce the detailed settings for the machine translation tasks and report the experimental results in this section.

4.1.1. Experimental Settings

Data We conduct experiments on standard translation tasks with multiple language pairs, which are English↔German (En↔De for short), English↔Dutch (En↔Nl for short), English↔Polish (En↔Pl for short), English↔Portuguese-Brazil (En↔Pt-br for short), English↔Turkish (En↔Tr for short), and English↔Romanian (En↔Ro for short) language pairs. These benchmark datasets all come from the widely acknowledged IWSLT-2014 machine translation (Cettolo et al., 2014) competition[2]. Each of them only contains 100k ∼ 200k data pairs, which can be viewed of low resource settings in some degree, since our approach works well in low resource learning tasks as introduced before. For each language pair, we work on bidirectional translations, i.e., En→X and X→En where X is another language, resulting in 12 machine translation tasks. Following (Bahdanau et al., 2016; Wu et al., 2017; 2018; Ott et al., 2019), we preprocess the datasets with some common acknowledged operations using Moses toolkit[3] (Koehn et al., 2007), such as lowercase words, length filtration. The resulted datasets contains about 160k, 7k and 7k pairs for training, valid and test sets for En↔De task, 180k, 4.7k, 1.1k for En↔Ro task, 170k, 4.5k, 1.1k for En↔Nl task, 175k, 4.5k, 1.2k for En↔Pt-br task, 181k, 4.7k, 1.2k for En↔Pl task and 160k, 4.5k, 1k for En↔Tr task respectively.

Tokenization We adopt the BPE[4] (Sennrich et al., 2015) and SentencePiece[5] (Kudo, 2018) tokenizers as previously introduced. The number of merge operations for BPE is 10k, resulting a near 10k subword vocabulary for all language pairs. For SentencePiece, we extract a 12k subword vocabulary for all language pairs. These vocabularies are built upon the joint source and target language sentences.

Model Since our model is based on the standard Transformer (Vaswani et al., 2017) architecture, we set the parameters for our model to be consistent with Transformer configurations for most ones. To be specific, we adopt `transformer_iwslt_de_en` configuration for the standard Transformer baseline models. The embedding dimension is set as 512 and the size of feed-forward layer is 1024. Each encoder and decoder contain 6 layers for each side. We share the source and target embeddings since we have a joint dictionary for each language pair. The dropout rate (Srivastava et al., 2014) is 0.3 and weight decay is 0.0001 for all experiments. The model is optimized with Adam (Kingma & Ba, 2014) optimizer and the learning rate schedule is the same default setting used as in Vaswani et al. (2017). Label smoothing (Pereyra et al., 2017) is also used with weight 0.1. In order to control the model parameters, we set 256 embedding size for our model instead of 512 used in the baseline model. We also share the parameters of mix-attention and self-attention components, as well as the BPE-enc-attention and SP-enc-attention components. That is, in Eqn.(1&2) and from Eqn.(4) to Eqn.(7), all layers with

[2] https://wit3.fbk.eu/mt.php?release=2014-01

[3] https://github.com/moses-smt/mosesdecoder/tree/master/scripts

[4] https://github.com/rsennrich/subword-nmt

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

Table 3. Machine translation results of our model and the standard Transformer on various IWSLT-2014 translation datasets. "Transformer_512" and "Transformer_256" refer to the baseline models with embedding dimension 512 and 256. Our models is equipped with embedding dimension 256. The numbers in bold font stand for results that are significantly better than Transformer_512 results with p-value less than 0.05 (Koehn, 2004).

	En→De		De→En		En→Ro		Ro→En		En→Nl		Nl→En	
	BPE	SP	BPE	SP	BPE	SP	BPE	SP	BPE	SP	BPE	SP
Transformer_512	28.80	28.45	34.84	34.77	24.56	24.67	32.07	31.72	29.23	29.48	33.33	33.08
Transformer_256	28.56	28.24	34.49	34.39	23.97	24.27	30.74	31.11	29.16	29.28	32.8	33.02
Our model	28.96	28.88	**35.51**	35.25	**25.11**	24.86	32.11	32.21	**29.85**	29.97	33.82	**33.96**
+ *co-teaching*	**29.93**	**29.71**	**36.41**	**36.31**	**25.98**	**25.68**	**33.12**	**32.89**	**31.16**	**31.11**	**34.45**	**34.61**
	En→Pt-br		Pt-br→En		En→Pl		Pl→En		En→Tr		Tr→En	
	BPE	SP	BPE	SP	BPE	SP	BPE	SP	BPE	SP	BPE	SP
Transforme_512	38.72	38.86	43.52	43.78	13.08	12.77	17.85	17.86	16.32	16.07	25.06	25.49
Transformer_256	38.44	38.53	42.57	42.88	12.73	12.73	17.33	17.12	15.62	15.93	24.74	24.90
Our model	39.19	39.08	43.98	43.44	13.46	**13.21**	18.07	17.79	16.49	**16.69**	**25.84**	25.68
+ *co-teaching*	**40.31**	**40.16**	**44.19**	**44.24**	**14.08**	**13.88**	18.26	**18.44**	**18.06**	**18.29**	**27.06**	**26.92**

the same superscript are shared. As for other configurations, we keep them to be same as the baseline settings. Since the mixed representations are added by two attention layers, to better balance the contribution of two inputs, following Larsson et al. (2016); Zhu et al. (2020), we add drop-path trick in both encoder and decoder, with drop value 0.2 and 0.3. The implementation is based on the Fairseq (Ott et al., 2019) toolkit[6].

Evaluation We use beam search (Sutskever et al., 2014; Medress et al., 1977) algorithm to generate the translation results. The beam width is 5 and the length penalty is 1.0. We report both BPE and SP results when testing the effectiveness of our model architecture and co-teaching algorithm. The evaluation metric is the commonly used tokenized BLEU (Papineni et al., 2002) score with `multi-bleu.perl` script[7].

4.1.2. Results

We report the BLEU results in Table 3. The Transformer baseline models are trained with embedding size 512 (denoted as Transformer_512) and 256 (denoted as Transformer_256). Our model is equipped with embedding size 256, which ensures that the model size does not exceed "Transformer_512". For example, in En↔De tasks, the number of parameters of the Transformer_512, Transformer_256, and our model are 37M, 13M and 27M respectively. As shown in Table 3, Transformer_512 baseline model with more parameters can lead to better results than Transformer_256 as expected.

With our proposed model, we can achieve consistent improvements across all the 12 translation tasks, no matter for the BPE tokenizer or the SP tokenizer. Compared with Transformer_256, our proposed model architecture outperforms the baseline by a non-trivial margin. For example, on De→En, En→Ro, Pt-br→En, we can obtain more than 1.0 BLEU score improvement. Compared with Transformer_512, both our BPE decoded and SP decoded results can outperform the corresponding baselines by 0.5 points for most tasks. The improvements demonstrate the effectiveness of mixed representations from two tokenizers, and our model architecture.

We then apply co-teaching to our model. On En↔De, we found that the generated sequences with BPE format or SP format are of similar quality, and BPE tokenizer can lead to slightly better results. We report both of the BPE and SP results in the table. We can see that on 11 out of 12 translation tasks, co-teaching makes more than 1.0 improvement. On En→Tr, we achieve additional more than 1.5 points improvement. That is, our model architecture is further benefited from co-teaching.

Comparison with existing works We summarize the recent results of several existing works on the widely acknowledged benchmark IWSLT De→En translation in Table 4, in order to give a clear comparison with our approach. These works vary from multiple aspects, e.g., the training algorithm design (Wang et al., 2019a), model architecture design (Lu et al., 2019; Wu et al., 2019) and data augmentation (Zhu et al., 2019). From the table, we can obverse that our approach outperforms all the previous works, which can

[6] https://github.com/pytorch/fairseq

[7] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

again demonstrate our method is very effective. In particular, our approach surpasses the BERT-fused NMT model (Zhu et al., 2020), which incorporates the large-scale pretrained BERT language model. Our method is complementary to previous works and we will further explore how to combine them together in the future.

Table 4. Comparison with existing works on IWSLT De→En translation tasks.

Approach	BLEU
Transformer_512 (Vaswani et al., 2017)	34.84
Adversarial MLE (Wang et al., 2019a)	35.18
DynamicConv (Wu et al., 2019)	35.20
Macaron (Lu et al., 2019)	35.40
Multi-Agent Dual Learning (Wang et al., 2019c)	35.56
Joint Self-Attention (Fonollosa et al., 2019)	35.70
Soft Contextual Data Aug (Zhu et al., 2019)	35.78
BERT-fused NMT (Zhu et al., 2020)	36.11
Ours	36.41

4.2. Abstractive Summarization

Text summarization is another widely acknowledged sequence generation task. There are two types of summarization tasks: Extractive summarization, where the summary is obtained by extracting sentences from the article; abstractive summarzation, where the summary is generated from scratch. In this work, we focus on the neural based abstractive summarzation.

Data We conduct the summarization experiment on a benchmark dataset, the Gigaword summarization dataset. The corpus is constructed from a subset of Gigaword corpus (Graff & Cieri, 2003). As described in Rush et al. (2015), the first sentence of an article serves as the source input, and the headline as target output. The preprocessing is identical to previous works (Shen et al., 2016; Rush et al., 2015). On average, each input article sentence contains about 31.4 words, and the corresponding summarization/headline consists of 8.3 words. The standard split data of article-headline pairs for training, validation and test set contain 3.8M, 190k and 1,951 samples respectively.

Tokenization Similar to the machine translation task, we also utilize the BPE and SP tokenizers to process the data. The vocabulary sizes of BPE tokenized corpus and SP tokenized corpus are 29,236 and 30,020 respectively

Model We choose the base Transformer architecture (`transformer_base` configuration) as the baseline. Each encoder and decoder are stacked by 6 blocks. We set the embedding dimension of our model as 256, the baseline models are also 512 and 256. The remaining hyperparameters are the same as those for machine translation tasks (see Section 4.1).

Table 5. Results of abstractive summarization.

BPE	R-1	R-2	R-L
Transformer_256	35.1	17.4	32.5
Transformer_512	35.5	17.5	32.7
Our model	35.7	17.8	33.0
+ *co-teaching*	36.0	18.4	33.4
SP	R-1	R-2	R-L
Transformer_256	34.2	17.2	32.1
Transformer_512	34.5	17.5	32.2
Ours	34.7	17.6	32.5
+ *co-teaching*	35.1	18.1	33.0

Evaluation In decoding process, we use beam search with width 4 and length penalty 1.0, where repeated trigrams are blocked (Paulus et al., 2017). The generation quality is evaluated by ROUGE (Lin, 2004) F1 score with an open source script[8]. Specifically, we report the unigram ROUGE-1 (R-1) and bigram ROUGE-2 (R-2) overlap to assess the informativeness and the longest common subsequence ROUGE-L (R-L) score to assess the fluency.

Results The results are shown in Table 5. When decoding into BPE formats, our model can boost the Transformer_256 baseline by 0.6, 0.4 and 0.5 point in terms of R-1, R-2 and R-L. After applying co-teaching, we can obtain more improvements, where the above three metrics can be further improved by 0.3, 0.6 and 0.4 point. Our method also outperforms the Transformer_512 baseline. Similar improvements are also observed when using SP tokenizer. These results demonstrate that our method generally works across different sequence generation tasks.

5. Ablation Study and Analysis

To better understand our model structure and training algorithm, we give ablation studies and analyses towards different aspects. The studies are conducted on the IWSLT-2014 En↔De machine translation tasks.

5.1. Stacked Representation

As discussed before, in the encoders of our proposed model, the mixed representations are obtained by adding outputs from mix-attention components and self-attention components. Different methods can be adopted to mix the representations of two tokenizers. Here we present an alternative approach: the stacked representations.

[8] https://github.com/pltrdy/pyrouge

The model structure to obtain stacked representations is shown in Figure 2. The mix-attention components process the output of self-attention components, instead of processing the input with self-attention components in parallel. The decoder is revised accordingly, where in $\texttt{dec}_{bpe}$, self-attention, SP-Enc-attention and BPE-Enc-attention are sequentially used, and self-attention, BPE-Enc-attention and SP-Enc-attention are sequentially leveraged in $\texttt{dec}_{sp}$[9].

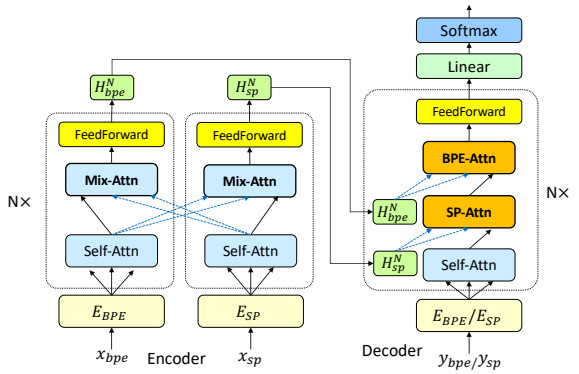

Figure 2. The model framework for stacked representation. Different from the mixed representations, the important components mix-attention (Mix-Attn in figure), SP-enc-attention (SP-Attn in figure) and BPE-enc-attention (BPE-Attn in figure) are stacked above the self-attention component instead of adding up. Note that we do not show residual connection and layer norm operation after each component in the figure, they are identical to the Transformer.

We evaluate the model performance of stacked representation without co-teaching. The results of En↔De translations are reported in Table 6. The stacked representations bring comparable results to Transformer baseline, both of which are worse than our mixed representations. We suspect the sequentially stacked version may underrate the representations from the self-attention itself and the overrated mix-attention confuses the final representation.

Table 6. Results of stacked representations on En↔De translations.

	En→De		De→En	
	BPE	SP	BPE	SP
Transformer_512	28.80	28.45	34.84	34.77
Transformer_256	28.56	28.24	34.49	34.39
Stacked	28.53	28.29	35.19	34.61
Ours	28.96	28.88	35.51	35.25

[9]We also evaluate the reversed order, mix-attention first and then self-attention component, the result is similar.

5.2. Co-Teaching with One Tokenizer

Though our co-teaching algorithm interacts between the two types of tokenized sequences, it can also be applied to one type only. That is, given a pre-trained model (standard Transformer), we use it to forward translate the data, then continue tuning the model on the newly generated data as well as the original data iteratively.

We conduct experiments on En↔De translations. The results are reported in Table 7. From the results, we can observe that co-teaching achieves significant improvements over the baseline models. This demonstrates that the co-teaching generally works across different settings. For En→De translation with SentencePiece, co-teaching with one tokenized input can obtain 29.28 BLEU score, while ours with two tokenized inputs can obtain 29.93 BLEU score. Similar results can be found for other settings. This shows the effectiveness of our interactive co-teaching between two tokenization ways.

Table 7. Results of co-teaching with only one type of tokenization on En↔De translations.

	En→De		De→En	
	BPE	SP	BPE	SP
Transformer_512	28.80	28.45	34.84	34.77
+ *co-teaching*	**29.67**	**29.28**	**36.10**	**35.71**

5.3. Other Tokenizations

Previous experiments are all conducted on BPE and SP subword tokenizations. Besides these two tokenizers, we also investigate the WordPiece (WP) tokenizer in this subsection. We replace the SP tokenizer to be WP and set the subword dictionary to be near 10k. Other configurations and training details are all same as those used in Section 4.1.

Table 8. Results of our model with BPE and WP tokenizers on En↔De translations.

	En→De		De→En	
	BPE	WP	BPE	WP
Transformer_512	28.80	28.71	34.84	34.91
Our model	**28.98**	**29.15**	**35.23**	**35.24**

Results are presented in Table 8. We can observe our model architecture also achieves better performance with BPE and WP tokenizers over the baselines, which again demonstrates that leveraging multiple tokenizers to build mixed representations is helpful to improve the translation quality.

6. Conclusion and Future Work

In this paper, we propose to generate sequences with mixed representations by leveraging different subword tokenization methods. Specifically, we introduce a new model structure to incorporate mixed representations from different tokenization methods, and a co-teaching algorithm to better utilize the diversity and advantage of each individual tokenization method. The effectiveness of our approach is verified on machine translation task and abstractive summarization application.

For future work, there are many possible directions. First, we will apply our algorithm to more sequence learning applications, like text classification, natural language understanding, etc. Second, we will further improve our model architecture, such as designing two interactive decoders, and adaptively determine which decoder we should use for inference when a sentence comes. Third, we will study to extend our model with more subword tokenization methods.

References

Arppe, A., Carlson, L., Lindén, K., Piitulainen, J. O., Suominen, M., Vainio, M., Westerlund, H., and Yli-Jyrä, A. M. *Inquiries into words, constraints and contexts: Festschrift in the honour of Kimmo Koskenniemi on his 60th birthday*. CSLI Publications, 2005.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., and Federico, M. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, volume 57, 2014.

Cherry, C., Foster, G., Bapna, A., Firat, O., and Macherey, W. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4295–4305, 2018.

Deng, L. and Liu, Y. *Deep learning in natural language processing*. Springer, 2018.

Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., et al. Recent advances in deep learning for speech research at microsoft. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8604–8608. IEEE, 2013.

Fonollosa, J. A., Casas, N., and Costa-jussà, M. R. Joint source-target self attention with locality constraints. *arXiv preprint arXiv:1905.06596*, 2019.

Graff, D. and Cieri, C. English gigaword, linguistic data consortium, 2003.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., et al. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*, 2018.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koehn, P. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 388–395, 2004.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. 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*, pp. 177–180, 2007.

Kudo, T. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.

Kudo, T. and Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

Larsson, G., Maire, M., and Shakhnarovich, G. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

Lee, J., Cho, K., and Hofmann, T. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378, 2017.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*, 2015.

Lu, Y., Li, Z., He, D., Sun, Z., Dong, B., Qin, T., Wang, L., and Liu, T.-Y. Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762*, 2019.

Luong, M.-T. and Manning, C. D. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1054–1063, 2016.

Medress, M. F., Cooper, F. S., Forgie, J. W., Green, C., Klatt, D. H., O'Malley, M. H., Neuburg, E. P., Newell, A., Reddy, D., Ritea, B., et al. Speech understanding systems: Report of a steering committee. *Artificial Intelligence*, 9 (3):307–316, 1977.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. 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)*, pp. 48–53, 2019.

Pan, Y., Li, X., Yang, Y., and Dong, R. Morphological word segmentation on agglutinative languages for neural machine translation. *arXiv preprint arXiv:2001.01589*, 2020.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Paulus, R., Xiong, C., and Socher, R. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

Provilkov, I., Emelianenko, D., and Voita, E. Bpe-dropout: Simple and effective subword regularization. *arXiv preprint arXiv:1910.13267*, 2019.

Rush, A. M., Chopra, S., and Weston, J. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 379–389, 2015.

Schuster, M. and Nakajima, K. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152. IEEE, 2012.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Shen, S., Zhao, Y., Liu, Z., Sun, M., et al. Neural headline generation with sentence-wise optimization. *arXiv preprint arXiv:1604.01904*, 2016.

Srinivasan, T., Sanabria, R., and Metze, F. Multitask learning for different subword segmentations in neural machine translation. *arXiv preprint arXiv:1910.12368*, 2019.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, D., Gong, C., and Liu, Q. Improving neural language modeling via adversarial training. In *International Conference on Machine Learning*, pp. 6555–6565, 2019a.

Wang, X., Pham, H., Arthur, P., and Neubig, G. Multilingual neural machine translation with soft decoupled encoding. *arXiv preprint arXiv:1902.03499*, 2019b.

Wang, Y., Xia, Y., He, T., Tian, F., Qin, T., Zhai, C. X., and Liu, T. Y. Multi-agent dual learning. In *7th International Conference on Learning Representations, ICLR 2019*, 2019c.

Wu, F., Fan, A., Baevski, A., Dauphin, Y. N., and Auli, M. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019.

Wu, L., Zhao, L., Qin, T., Lai, J., and Liu, T.-Y. Sequence prediction with unlabeled data by reward function learning. In *IJCAI*, 2017.

Wu, L., Tian, F., Zhao, L., Lai, J., and Liu, T.-Y. Word attention for sequence to sequence text understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.

Zhu, J., Gao, F., Wu, L., Xia, Y., Qin, T., Zhou, W., Cheng, X., and Liu, T.-Y. Soft contextual data augmentation for neural machine translation. *arXiv preprint arXiv:1905.10523*, 2019.

Zhu, J., Xia, Y., Wu, L., He, D., Qin, T., Zhou, W., Li, H., and Liu, T. Incorporating bert into neural machine translation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hyl7ygStwB.