Local Byte Fusion for Neural Machine Translation

Makesh Narsimhan Sreedhar[1], **Xiangpeng Wan**[2], **Yu Cheng**[3], **Junjie Hu**[1]
[1]University of Wisconsin-Madison, [2]NetMind.AI and ProtagoLabs, [3]Microsoft Research
{msreedhar,junjie.hu}@wisc.edu

Abstract

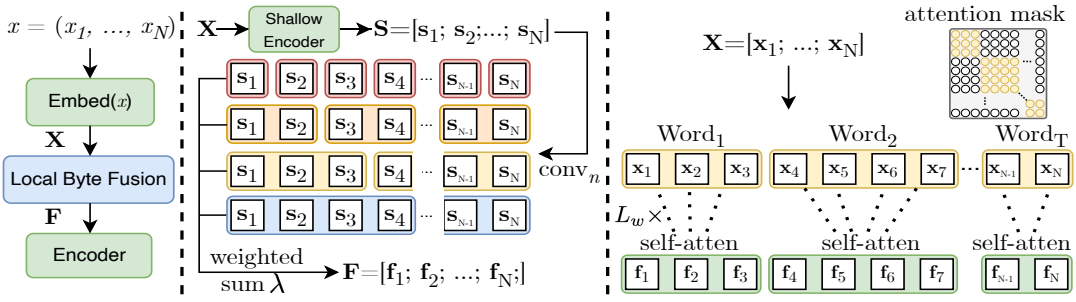
Subword tokenization schemes are the dominant technique used in current NLP models. However, such schemes can be rigid and tokenizers built on one corpus may not adapt well to other parallel corpora. It has also been observed that in multilingual corpora, subword tokenization schemes oversegment low-resource languages, leading to a drop in translation performance. An alternative to subword tokenizers is byte-based tokenization, i.e., tokenization into byte sequences using the UTF-8 encoding scheme. Byte tokens often represent inputs at a *sub-character* granularity, i.e., one character can be represented by a span of byte tokens. This results in much longer byte sequences that are hard to interpret without aggregating local information from multiple byte tokens. In this paper, we propose a **Lo**cal **B**yte **F**usion (LOBEF) method for byte-based machine translation—utilizing byte n-gram and word boundaries—to aggregate local semantic information. Extensive experiments on multilingual translation, zero-shot cross-lingual transfer, and domain adaptation reveal a consistent improvement over vanilla byte-based models. Further analysis also indicates that our byte-based models are parameter-efficient and perform competitive to subword models.

1 Introduction

Multilingual neural machine translation (NMT) has proven effective to transfer knowledge learned from a high-resource language to a low-resource language. However, existing multilingual NMT models still rely on a pre-built subword tokenizer (e.g., BPE (Sennrich et al., 2016), SentencePiece (Kudo and Richardson, 2018)) to tokenize a sentence into a sequence of subword units. This has two drawbacks. First, once the tokenizer is fixed, we lose the flexibility of changing the word tokenization if we aim to fine-tune the NMT model on another parallel corpus of interest for adaptation. Second, when a subword tokenizer is built on unbalanced multilingual data, word tokens from a low-resource language are usually under-represented, resulting in over-segmentation of a word into many single characters. A recent study (Rust et al., 2021) measures over-segmentation by the *fertility* score of a subword scheme which indicates how many subwords a whole word is broken down into. It shows a negative correlation between the fertility score and the performance of multilingual models on the over-segmented languages.

Although character-based models have often been proposed as a solution to these problems (Gupta et al., 2019; Libovický and Fraser, 2020; Li et al., 2021), they come with their own tradeoffs related to the significant overhead of processing long character sequences during training and inference (Libovický and Fraser, 2020). Besides, these models still adopt a fixed vocabulary of characters, leading to the same issue as a fixed subword tokenizer for adaptation. Another point of difference is that in character-based methods, the vocabulary still consists of one unique embedding for each character under consideration. In byte-based approaches, the tokens are at a sub-character granularity and the model has to figure out how to combine bytes for different languages. Recently, there has been renewed interest in character-based models that adopt a byte tokenization scheme (Clark et al., 2021; Xue et al., 2021; Tay et al., 2021)—tokenization of texts into UTF-8 byte tokens. Although these byte-based models have shown competitive performance to subword models on multilingual NLU benchmarks (Hu et al., 2020), their performance on multilingual generation, especially on multilingual NMT is still underexplored. Although Shaham and Levy (2021) recently demonstrate the effectiveness of byte tokenization for bilingual machine translation, a comprehensive study of such byte-based methods on the multilingual paradigm across a wide variety of languages and domains is still

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7199–7214



(a) LOBEF (b)Byte-nCF (c)Byte-WSF

Figure 1: (a) LOBEF; (b) **Byte-nCF** uses four convolutional layers (width=n, stride=n) to aggregate char-level information; (c) **Byte-WSF** uses word boundaries with block-wise self-attention to aggregate word-level information.

missing. Particularly in the multilingual setting, as characters in different languages can be tokenized into a varying number of byte tokens, this produces byte sequences much longer than the original sentences, and vanilla byte-based MT models can only *implicitly* reconstruct character-/word-level representations from byte tokens in an entirely data-driven fashion.

To remedy these issues, we propose two variants of **Lo**cal **Byt**e **F**usion (LOBEF) [1] techniques that explicitly aggregate byte tokens to learn character-/word-level representations for byte-based NMT models. Our first variant utilizes four n-gram convolutional layers to aggregate bytes for learning character-level information, and our second variant utilizes word boundaries to aggregate a span of bytes for learning word-level context. We conduct extensive experiments to compare our methods with the vanilla byte-based model and the embeddingless model from Shaham and Levy (2021) in a multilingual translation setting. Our many-to-one translation results show that aggregating local information in earlier layers encourages the model to capture local information for seven source languages, yielding an average gain of up to 1.4 BLEU over the vanilla byte-based NMT model while performing competitively with subword models. We further demonstrate the effectiveness of LOBEF on the zero-shot/few-shot cross-lingual transfer and cross-domain adaptation settings, showing the flexibility of byte-based NMT models over subword baselines when fine-tuning is required for data adaptation. Additionally, our method also improves over vanilla byte-based NMT models for adaptation. Our contributions are as follows:

- To the best of our knowledge, we are the first to evaluate byte-based embeddingless NMT models in a multilingual translation setting.
- To further improve the encoding of local semantics for byte-based NMT, we propose two variants of local fusion techniques based on character-/word-level aggregation over byte tokens.
- We provide a fine-grained analysis to show the effectiveness of byte-based models on cross-lingual and domain adaptation settings.

[1]The code can be found at https://github.com/makeshn/LOBEF_Byte_NMT

2 Preliminaries

2.1 Unicode and UTF-8

Unicodeis a universal, platform-agnostic standard for handling text in most of the world's writing systems, covering characters in all of the world's living languages as well as emoji and non-visual codes. Each code point defined by Unicode is mapped to a unique integer, ranging from 0 to 10FFFF$_{16}$. For instance, the English character set *A-Z* is denoted by the integers from *97-122*. In modern computers, each Unicode code point can be implemented as bytes by multiple encoding protocols, and UTF-8 is the dominant encoding protocol used by over 95% of webpages.

In UTF-8, each Unicode code point is represented as one to four bytes (8 bits per byte) depending on the range of its Unicode integer. Some languages may have a combination of characters that require a varying number of bytes. For example, most characters in German require only a single byte, while some special characters like $\grave{a}$ or $\hat{a}$ use two bytes. Since the Unicode and the UTF-8 encoding scheme is already well-defined, we do not have to construct source and target vocabularies similar to how it is done for subword models. Tokenization and de-tokenization for byte based models is as simple as a single line of code in Python and does not involve any heuristic preprocessing. In this paper, we adopt the UTF-8 byte

tokens as inputs to our model.

2.2 Byte-based NMT

Shaham and Levy (2021) recently propose an *embeddingless* NMT model that takes sequences of UTF-8 byte tokens as the inputs and outputs, and uses a fixed one-hot representation for each byte token instead of a dense learnable embedding vector. Such a byte-based NMT model eliminates the input and output token embedding layers usually used in subword-based NMT models, leading to a significant reduction in model parameters.

Formally, given a source-target sequence pair from a parallel corpus $(x, y) \sim \mathcal{D}$ where $x = (x_1, ..., x_N)$ and $y = (y_1, ..., y_M)$ are both sequences of byte tokens, the input sequence is first embedded by one-hot representations, i.e., $\text{Embed}(x) = \mathbf{X} \in \mathbb{R}^{N \times d}$, and further encoded into the source hidden representation $\mathbf{Z}$ by a vanilla L-layer Transformer encoder:

$$\mathbf{Z} = \text{Encoder}(\mathbf{X}, L). \quad (1)$$

Finally, an attention-based decoder performs the attention over $\mathbf{Z}$ and estimates the probability of predicting the next byte token y_t by

$$P(y_t | y_{<t}, x) = \text{Decoder}(y_{<t}, \mathbf{Z}). \quad (2)$$

Compared to subword-based NMT models, byte-based NMT models have shown effectiveness on bilingual machine translation, while their performance in multilingual machine translation is still unexplored. Especially in the many-to-one translation, the encoder is used to encode multiple languages which aggregate varying numbers of byte tokens (i.e., 1 to 4 bytes) to represent one character.

3 Local Byte Fusion

For languages that do not exclusively use the English character set, encoding them often requires more than one byte. Vanilla byte-based models can only *implicitly* aggregate character-level or word-level representations for these languages, potentially resulting in poor interpretability and sub-optimal results in multilingual settings. Hence, we propose two fusion techniques that encourage models to *explicitly* aggregate character-level and word-level information from byte sequences.

We also adopt byte sequences as inputs and outputs for our model, and use vanilla Transformer as the backbone. As we focus on multilingual encoding in this work, we only modify the encoder, and adopt the same decoder architecture from (Shaham and Levy, 2021). Note that a more sophisticated design of the decoder will also involve a special design of decoding algorithms (Libovický et al., 2022) which goes beyond the scope of this work. Besides, to conduct a more comprehensive study, we also consider the case where we retain embedding layers for the encoder and decoder of the byte-based model. This implies that instead of one-hot representations for the byte sequences, we can learn dense vector representations. Since the vocabulary size of all byte tokens is 256, this does not amount to adding a significant number of extra parameters.

3.1 n-gram Convolutional Fusion (nCF)

Before we explicitly aggregate the character-level information, we first encode the input byte sequence by a shallow encoder with L_s Transformer layers, which allows the model to have a shallow access to the sentence context before local fusion.

$$\mathbf{S} = \text{Encoder}(\mathbf{X}, L_s) \quad (3)$$

Since characters can be represented as a combination of 1 to 4 bytes depending on the languages, we apply four different 1-D convolutional layers to aggregate the n-gram byte tokens where $n \in \{1, 2, 3, 4\}$. Specifically, we define conv_n as the 1-D convolution layer with a kernel of size n and stride n. We do right padding at the end of the byte sequence. Therefore, when we use a stride n greater than 1, the length of the input byte sequence is reduced by a factor corresponding to the stride n. We then define the output from the conv_n layer by:

$$\left[\mathbf{f}_1^n, \cdots, \mathbf{f}_{\frac{N}{n}}^n\right] \in \mathbb{R}^{\frac{N}{n} \times d} \leftarrow \text{conv}_n(\mathbf{S}). \quad (4)$$

To make all of the outputs the same length as the input, we repeat the output tokens in place by a factor corresponding to the stride length n.

$$\mathbf{F}^n = \left[\text{repeat}(\mathbf{f}_1^n, n), \cdots, \text{repeat}\left(\mathbf{f}_{\frac{N}{n}}^n, n\right)\right], \quad (5)$$

where $\text{repeat}(\mathbf{x}, n)$ creates n copies of a vector $\mathbf{x}$. Applying this repetition process to the output from each of the convolution layers, we have four representations of equal sequence length as the source sequence,[2] i.e., $\mathbf{F}^1, \mathbf{F}^2, \mathbf{F}^3, \mathbf{F}^4 \in \mathbb{R}^{N \times d}$. We pass these representations through a linear layer to get a

[2]Extra tokens at the end are truncated to ensure equal length.

single weighted representation:

$$\mathbf{F} = \sum_{n=1}^{4} \lambda^n \mathbf{F}^n, \quad (6)$$

where $\lambda = [\lambda^1, \cdots, \lambda^4]$ are weights for the n-gram representations. We pass this weighted representation to the remaining $(L - L_s)$ Transformer layers to obtain the final encoder hidden representation which is further sent to the decoder by Eq. (2).

$$\mathbf{Z} = \text{Encoder}\left(\mathbf{F}, L - L_s\right) \quad (7)$$

The n-gram fusion enables the model to learn what combination of the input byte sequence representations results in better character-level features.

3.2 Word-based Self-attention Fusion (WSF)

In addition, we also propose a word-based self-attention fusion method that utilizes the word boundary information in the raw sentence to aggregate byte tokens within the same word. As characters in most languages are represented by more than one byte and words contain varying number of characters, using byte tokens as input to the model results in a much longer sequence. Therefore, this property may require the model to recognize a meaningful span of byte tokens in order to capture the semantic of a word token in the raw sentence. However, vanilla byte-based NMT models (§2.2) use the traditional *full self-attention*, which implies that every byte token in the sequence attends to all byte tokens even though some far-away byte tokens may have little association to the query byte. Besides, as words are represented by a span of bytes in a small vocabulary of size 256, it is likely to produce a high attention weight between two identical byte tokens even when these byte tokens are used in two completely irrelevant words.

We tackle this issue by aggregating local information of a byte span for a word using a *block-wise* self-attention. Formally, for a byte sequence $x = (x_1, \cdots, x_N)$, we define its (untokenized) word sequence as $w = (w_1, \cdots, w_T)$ and a mapping $\pi : [t] \rightarrow [a : b]$ that maps the word index t to the beginning and the end indices of the corresponding byte span, i.e., $w_t = x_{\pi(t)} = x_{a:b}$. By leveraging the word boundary, we naturally break the long byte sequence into a list of sub-sequences, then we apply an L_w-layer Transformer encoder to encode byte tokens only in their sub-sequences:

$$\mathbf{F}_{\pi(t)} = \text{Encoder}(x_{\pi(t)}, L_w), \forall t \in [1 : T], \quad (8)$$

where $\mathbf{F}_{\pi(t)} \in \mathbb{R}^{|b-a| \times d}$ is hidden representation of byte tokens in the t-th word spanning over the sub-sequence $x_{a:b}$. This allows byte tokens to effectively aggregate local information for each word token, which is useful for the model to distinguish identical byte tokens used in two different words. Note that the word-based self-attention in Eq. (8) can be efficiently implemented by pre-computing a *block-wise* attention mask matrix (Figure 1 (c)), ensuring that self-attenion is only performed among a byte span of a word in a Transformer layer. Finally we obtain the *word-aware* representation of the input byte sequence $\mathbf{F}$ by putting $\mathbf{F}_{\pi(t)}$ in the word order, i.e., $\mathbf{F} = [\mathbf{F}_{\pi(1)}, \cdots, \mathbf{F}_{\pi(T)}] \in \mathbb{R}^{N \times d}$, and feed $\mathbf{F}$ as input to the remaining $(L - L_w)$ Transformer layers similar to Eq. (7).

4 Experimental Settings

4.1 Datasets

Multilingual Many-to-One Translation: We use the OPUS public data (Tiedemann, 2012) to construct a multilingual parallel corpus that has a fair mix of high-resource and low-resource languages. We train a multilingual translation model from seven source languages to English. Table 1 shows the statistics of the training data. We use the Flores-101 (Goyal et al., 2022) benchmark to evaluate the performance of our models.

Zero-shot Cross-lingual Translation: Following Neubig and Hu (2018), we use the same Ted Talk dataset that include four language pairs where each pair has a high-resource language (HRL) and a low-resource languages (LRL) written in the same script. Table 2 shows the statistics of the dataset.

Cross-domain Adaptation: In this task, we train all models on the WMT19 German-English dataset on the news domain, and directly evaluate on the test data used in Aharoni and Goldberg (2020) from three diverse domains (Koran, IT, Medical).

4.2 Models

To fully evaluate the efficacy of byte-based techniques, we consider models under settings where we learn *dense embeddings* for the input byte tokens (DENSE) as well as the *embeddingless* case where there are no learnt embeddings (ONE-HOT). Our main baseline for comparison is the vanilla byte-based model and the ONE-HOT model proposed in Shaham and Levy (2021). We also include

Lang.	ID	Script	Fertility	#Train	#Test
German	deu	Latin	1.6	2.56M	1,012
Hindi	hin	Devanagari	1.6	1.6M	1,012
Nepali	npi	Devanagari	2.0	445K	1,012
Tamil	tam	Brahmic	2.6	268K	1,012
Telugu	tel	Brahmic	2.5	108K	1,012
Khmer	khm	Khmer	8.5	127K	1,012
Lao	lao	Lao	9.5	2.7K	1,012

Table 1: Writing scripts, fertility of seven source languages, no. of sentences in the many-to-English training set from OPUS and test set from Flores-101.

results of subword and character-based models for a holistic comparison.

Subword Model: We use BPE models trained using `Sentencepiece`[3] as our subword model.

Char Model: We use character-based models with inputs and outputs being character sequences.

Byte Based Models: For each of these models, we consider both ONE-HOT variants where models do not have learnt embeddings and DENSE variants where we learn continuous dense embeddings.

- **Byte**: Similar to the vanilla byte-based model proposed by (Shaham and Levy, 2021), the inputs and outputs are UTF-8 byte tokens.
- **Byte-nCF**: We use a shallow Transformer encoder[4] ($L_s = 1$) and four convolutional layers to fuse character-level information and learn a weighted *n-gram* representation (§3.1)
- **Byte-WSF**: We use a L_w-layer Transformer encoder[5] with a word-based self-attention over byte tokens within word boundaries (§3.2).

4.3 Multilingual Translation

In this experiment, we evaluate the subword and byte-based models on many-to-one translation (**xx-eng**) where **xx** refers to seven source languages listed in Table 1. We first clean the training data by removing sentences that are longer than 800 bytes in either the source or the target side, and then tokenize the sentences using the `Moses` tokenizer.[6] Doing such preprocessing does not affect the diversity of the dataset in terms of length as less than 0.5% of the samples are discarded. The byte-based models do not have any preprocessing apart from the `Moses` tokenization and even whitespaces are included as valid tokens. For low-resource languages that share the same script as high-resource languages, we can reuse the same tokenizer for the high-resource language. For the subword-based model, we construct a shared vocabulary of 64K BPE tokens for all the source languages and an English vocabulary of 8K BPE tokens for this experiment. All models are trained for the same number of epochs on our OPUS train set, and evaluated on the Flores-101 test set.

4.4 Cross-lingual Transfer

This experiment evaluates how effective subword and byte-based methods are in transferring performance across languages that share similar language scripts. We train both the subword and byte-based models on parallel data in a high-resource language (HRL) for 50K steps, and evaluate them in a zero-shot manner on the corresponding low-resource language (LRL) without training on any LRL data. Table 2 shows the data statistics. We focus on **xx-eng** translation where **xx** is either HRL or LRL.

In the case of subword models, this amounts to constructing a vocabulary (i.e., BPE tokenizer) based on only the HRL data and using that to tokenize the LRL data, while byte-based models use an universal tokenization scheme to tokenize both HRL and LRL data into UTF-8 byte tokens.

We also investigate a few-shot setting where the models pre-trained on the HRL data is further fine-tuned on a few parallel training samples in LRL. We examine the impact of different numbers (i.e., 1K, 2K, 3K, and 4K) of few-shot samples on the translation performance of these models. We fine-tune all models for 5K steps on the few-shot samples, and then evaluate them on the test set in LRL.

HRL			LRL			
Language	ID	Train Size	Language	ID	Test Size	<unk>%
Turkish	tur	182k	Azerbaijani	aze	1k	41.5%
Russian	rus	208k	Belarusian	bel	0.6k	48.1%
Portugese	por	185k	Galician	glg	1k	23.7%
Czech	ces	182k	Slovak	slk	2.5k	30.0%

Table 2: Sentence sizes for LRL/HRL, and unknown token rate on the LRL test set using HRL BPE tokenizers

4.5 Cross-domain Adaptation

Having translation systems adapt to domains apart from the one it has been trained on is a good measure of how robust models are. In the experiment, we compare subword and byte-based models on how effectively they translate sentences from do-

[3] https://github.com/google/sentencepiece
[4] Appendix C shows that Byte-nCF works best on {deu,khm}-eng translations when fusing lower-layer representations.
[5] Appendix D shows that $L_w = 4$ empiricall works best.
[6] https://github.com/moses-smt/mosesdecoder

Lang. Pairs		DENSE Models					ONE-HOT Models		
Src	Tgt	Subword	Char	Byte	Byte-nCF	Byte-WSF	Byte	Byte-nCF	Byte-WSF
deu	eng	31.5	29.2	31.3	**32.1**	31.7	31.1 (-0.4)	**31.6** (-0.6)	31.3 (-0.3)
hin	eng	24.8	21.9	23.9	**25.5**	25.4	24.3 (-0.2)	**25.6** (+0.1)	24.9 (+0.0)
npi	eng	18.1	17.3	18.2	19.8	**19.9**	17.9 (-0.1)	19.1 (-0.7)	**19.2** (-0.8)
tam	eng	18.2	17.4	17.9	**19.5**	18.9	18.3 (-0.4)	**19.1** (-0.4)	18.8 (-0.1)
tel	eng	20.3	17.5	20.5	**22.2**	22.1	19.9 (-0.5)	21.1 (-1.1)	**20.8** (-1.0)
khm	eng	12.6	11.1	12.4	13.5	**13.9**	12.2 (-0.2)	**12.6** (-1.3)	13.1 (-0.8)
lao	eng	**9.2**	7.7	5.9	6.4	6.5	5.8 (+0.1)	6.3 (-0.1)	**6.9** (+0.4)
Avg.		19.2	17.4	18.6	19.9	19.7	18.5 (-0.1)	19.4 (-0.5)	19.3 (-0.4)

Table 3: BLEU scores of the DENSE and ONE-HOT models on the Flores-101 dataset. Highest scores for each language pair on these two sets of models are highlighted in bold font. The differences of BLEU scores between ONE-HOT models and their corresponding DENSE variants are highlighted in the brackets.

mains that are not part of the training set. Similar to the cross-lingual transfer setting (§4.4), we train both subword and byte-based models on the source domain (News) and evaluate them in a zero-shot manner on three target domains (Koran, IT, Medical). Each model is trained on the source domain dataset for 50K steps, and then evaluated on each of the target domain test sets.

4.6 Hyperparameters

We use the `Fairseq`[7] library as the codebase. To make a fair comparison, we strictly follow the architectural choice of Shaham and Levy (2021) and employ the vanilla transformer encoder-decoder architecture as our backbone for all experiments. For all models, we use a total of 6 Transformer layers for the encoder and 6 layers for the decoder with 8 attention heads, 512 hidden units and the feed-forward dimension of 2048. We use the Adam(Kingma and Ba, 2014) optimizer with an inverseq square root learning rate scheduler, and warm up 4K steps to reach a peak learning rate of 5e-4. We apply a weight decay of 1e-4 and a label smoothing of 0.1. We also train all models for an equal number of epochs in all the experiments.

4.7 Evaluation

For a fair, consistent evaluation, we follow Shaham and Levy (2021) in using `Sacre-BLEU`[8] with 13a tokenizer for all language pairs using the raw text to compute the BLEU scores.

5 Results

In this section, we detail the results of the various experiments and discuss their implications.

[7] https://github.com/facebookresearch/fairseq

[8] https://github.com/mjpost/sacrebleu

5.1 Multilingual Translation

Table 3 shows the BLEU scores on the test set of the FLORES-101 data for many-to-one translation. We further investigate the following questions.

Do we need dense embeddings? In line with the findings of (Shaham and Levy, 2021) that embeddingless models are competitive with subword models in bilingual settings, we find that they perform on par with their corresponding models that use dense embeddings with an average difference of less than 0.5 BLEU over seven languages. We find that for six out of the seven source languages under consideration, the byte-based models perform competitively with the subword models. However, subword models still hold the edge for extremely low-resource languages such as *Lao-English* translation with only 2.7K training data. Besides, as Lao's written script is not shared with any of the other languages, we hypothesize that training byte-based multilingual models requires more training data in order to figure out the different fusion of byte tokens across languages while subword models with an explicit vocabulary for all languages do not have this requirement.

How effective is character/word fusion? Our proposed methods (Byte-nCF and Byte-WSF) that induce higher level semantic representations for bytes improve over vanilla byte-based models in both cases (ONE-HOT and DENSE models) on all language pairs with an average gain of up to 1.4 BLEU. Since sequence lengths tend to be extremely long when using byte sequences, aggregating information locally in the lower layers enables the model to quantitatively obtain higher scores than even subword-based models except in the extremely low-resource regime.

Which fusion works better? Comparing our two proposed variants, the Byte-nCF model per-

Lang. Pair Src	Tgt	DENSE Models Subword	Byte	Byte-nCF	ONE-HOT Models Byte	Byte-nCF
aze	eng	3.7	6.9	**7.8**	5.7	6.9
bel	eng	1.7	3.9	**5.3**	4.6	5.4
glg	eng	7.6	15.2	16.7	16.4	**17.2**
slk	eng	2.9	11.4	**12.6**	11.9	12.2
Avg.		4.0	9.4	**10.6**	9.7	10.4

Table 4: BLEU scores of the DENSE and ONE-HOT models on the Ted Talk dataset. Highest score among all models is in bold font.

forms slightly better than the Byte-WSF model in the DENSE case, while both perform comparably in the ONE-HOT case. In particular, Byte-nCF performs better than Byte-WSF on relatively high-resource languages (e.g., German, Hindi) with more than 1M training data. Besides, both variants perform comparably on low-resource languages (e.g., Khmer, Lao) with large fertility scores.

5.2 Cross-lingual Transfer

The performance of byte-based and subword models on cross-lingual transfer is shown in Table 4. As Byte-WSF and Byte-nCF have shown comparable performances in Table 3, we only include the Byte-nCF variant in the comparison below.

Does universal tokenization work? When evaluating subword and byte-based models in a zero-shot setting (§4.4), byte-based models outperform subword baselines by a clear margin of up to 6.1 average BLEU over all languages. The gains compared to vanilla byte baseline is 1.2 BLEU for DENSE variant and 0.7 BLEU for ONE-HOT models. Our results indicate that even for languages written in the same script, a rigid subword schedule is infeasible for NMT models to perform an effective cross-lingual transfer. Particularly, we observe a significant increase in BLEU in the glg-eng and slk-eng translations when using byte tokens as inputs.

Does fusion help cross-lingual transfer? We find that using the Byte-nCF fusion variant leads to marginal improvement over the vanilla byte-based model with an average gain of up to 0.4 BLEU. It should be noted that most of these language pairs share the same script and hence the convolution fusion technique works very well. Investigating whether such fusion techniques work for languages that do not share the same script can be explored in future work.

Domain	DENSE Models Subword	Byte	Byte-nCF	ONE-HOT Models Byte	Byte-nCF
WMT19 News	17.6	21.2	21.3	21.1	**21.5**
Koran	1.8	6.6	**7.4**	6.8	7.7
IT	3.9	10.4	**11.6**	10.2	11.3
Medical	4.1	13.6	15.3	12.9	**15.4**

Table 5: BLEU scores of the DENSE and ONE-HOT models on zero-shot cross-domain adaptation.

Does the few-shot setting improve performance? Figure 2 shows the translation performance in terms of BLEU for the BPE and byte-based models in the few-shot setting. We find that as the number of training data in LRL increases, the performance of the byte-based models improves, and Byte-nCF consistently improves over the vanilla byte model. The BPE baseline suffers from the issue of having a high unknown token rate and cannot take full advantage of the additional training data.

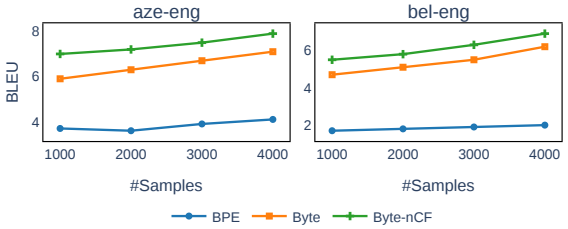


Figure 2: Few shot translation performance of BPE and Byte (ONE-HOT) based models.

5.3 Cross-Domain Adaptation

Table 5 shows the results of subword and byte-based models on zero-shot cross-domain adaptation. The first row indicates the BLEU scores on the *in-domain* test set (WMT19 News), and the other rows showcase the performance on *out-of-domain* test sets (Koran, IT, Medical).

Are byte-based models robust to domain shift? We find that the performance of both subword and byte-based models is susceptible to domain shifts. The BLEU scores on other domains are significantly lower for all variants. However, on comparison, we find that byte-based models are more robust than subword models against domain, yielding higher BLEU scores on all out-of-domain test sets.

Employing convolution fusion with the byte-based models improves performance over subword-based models, especially in the IT and medical domains. The issue with cross-domain adaptation remains that each new domain consists of specific jargon and entities that are not captured in the source

domain. This inhibits the models from capturing the required semantic information to translate out-of-domain sentences effectively.

6 Discussion and Analysis

Next, we further present a qualitative analysis of byte-based models and our proposed variants.

We use the `compare-mt` toolkit (Neubig et al., 2019) to holistically analyze how the outputs of these models differ and what aspects different models excel at. We compare the multilingual NMT models (§5.1) on the *German-English* translation as a sample language pair for the analysis. Specifically, we group all source words in the test sentences into buckets by the word fertility score (Figure 4a) and the word length in terms of characters (Figure 4b). Recall that the word fertility score measures how many subword units a word is broken into, and we use the BPE tokenizer used in Section 5.1. We evaluate byte-based models (i.e., Byte, Byte-WSF, Byte-nCF) using one-hot representations on each group in terms of source word translation accuracy.

As German is a high-resource language, most German words (45%) have a fertility score of 1 implying that they are not segmented by the BPE tokenizer, and all byte-based methods perform comparable on these words. We find that the Byte-nCF method performs better than the other two byte-based methods on oversegmented words (as indicated by the accuracy on words with fertility scores above 4). We also find that the Byte-nCF method outperforms other methods on translating long words (depicted by the accuracy on words with length greater than 15 characters). Comparing to the word-based (Byte-WSF) or sentence-level full self-attention (Byte), we hypothesize that this is a result of encoding a smaller sequence length when using the convolution fusion operation, reducing the pressure of byte-based models to capture too much information from a span of byte tokens.

7 Related Work

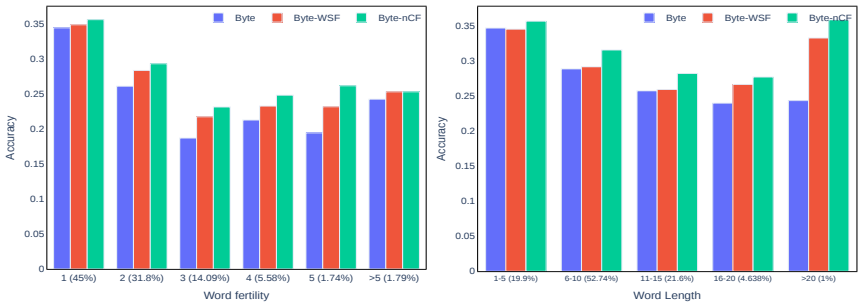
Subword Models Byte Pair Encoding (Sennrich et al., 2016), Wordpiece (Wu et al., 2016) and SentencepPiece (Kudo and Richardson, 2018) are widely-used subword tokenization schemes for NMT models, or perhaps most neural NLP models. However, the rigid tokenization scheme poses challenges in terms of oversegmenting low-resource languages and adapting a pre-trained model to new languages or new domains of different corpora (Sun et al., 2020; Bostrom and Durrett, 2020; Provilkov et al., 2020; Kudo, 2018; Godey et al., 2022).

Character-level Models Applying neural models directly on character sequences has been extensively studied (Sutskever et al., 2011; Graves, 2013; Kalchbrenner et al., 2016; Zilly et al., 2017; Melis et al., 2018; Al-Rfou et al., 2019; Kim et al., 2016; Gao et al., 2020; Tay et al., 2022). *Character-aware* methods were mainly developed by the use of word boundaries and convolutions over characters (Kim et al., 2015; Jozefowicz et al., 2016; Peters et al., 2018; El Boukkouri et al., 2020; Ma et al., 2020). However, for machine translation, character-based NMT models (Lee et al., 2017; Cherry et al., 2018; Libovický et al., 2022) still suffer from a high computational overhead of encoding and decoding much longer sequences.

Tokenization-free Methods Recent attempts have focused on using Unicode or UTF-8 encoding scheme to remove pre-built subword tokenizers from preprocessing. These byte-based methods have achieved promising results in terms of accuracy and speed-up in several multilingual language understanding tasks (Clark et al., 2021; Tay et al., 2021; Xue et al., 2021) or bilingual translation tasks (Shaham and Levy, 2021), while their application to multilingual or cross-lingual/domain settings is still underexplored.

8 Conclusion

We propose a **Lo**cal **B**yt**e** **F**usion (LOBEF) method with two variants—one employing convolutions on byte n-grams and the other utilizing word-based self-attention restricted to word boundaries. We show that these two fusion variants improve upon vanilla byte-based models indicating that neural machine translation models benefit from the explicit aggregation of local semantic information for characters or words at lower layers of neural networks. Our experiments show that both ONE-HOT and DENSE versions of byte-based models perform competitively on multilingual machine translation and even beat subword baselines on multiple language pairs. We also conduct an investigation of the effectiveness of byte-based techniques in both zero-/few-shot cross-lingual transfer and domain adaptation settings, and find that they outperform subword models by a large margin.



(a) Fertility Score vs Word accuracy (b) Word Length vs Word accuracy

Figure 3: Translation Word Accuracy grouped by word fertility/length for ONE-HOT Byte models on deu-eng.

9 Limitations

Despite achieving high translation performance on various language pairs, LOBEF has some limitations, coming from the nature of processing UTF-8 byte sequences.

Speed: As shown in Table 8 in the appendix, the inference times for byte-based models are higher when compared to subword-based models. It is also worth noting that we use the same amounts of model parameters for a total of 6 Transformer encoder layers and 6 Transformer decoder layers for all models in comparison. As shown in Table 7, byte-based models can effectively reduce the amounts of parameters for the embedding layers comparing to the subword-based models, leading to faster training time as shown in Table 8. However, as indicated by Xue et al. (2021), by adding more encoder layers, we can construct byte-based models with comparable amounts of parameters as subword-based models, and these larger byte-based models still require much longer time for training than subword-based models.

Extremely Low-resource Languages: The performance of byte-based models on extremely low-resource languages (e.g., 2.7K training data for Lao-English) is still lower than subword models especially in the multilingual setting. We suspect that byte-based methods require a relatively larger number of training data in order to aggregate information from a combination of byte tokens, comparing to subword-based models that explicitly maintain a subword vocabulary.

Extra Preprocessing: The Byte-WSF model requires an extra preprocessing step that precomputes the attention mask corresponding to the words in each sentence. This adds a slight overhead before training, while training the Byte-WSF model is as fast as the Byte model, as both model use the same Transformer architecture. However, for languages (e.g., Chinese) that do not have white-spaces to indicate the word boundary, we may rely on an off-the-shell word segmentation tool to preprocess the text.

References

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763, Online. Association for Computational Linguistics.

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3159–3166.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. 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. Association for Computational Linguistics.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2021. Canine: Pre-training an efficient tokenization-free encoder for language representation.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. 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). International Committee on Computational Linguistics.

Yingqiang Gao, Nikola I. Nikolov, Yuhuang Hu, and Richard H.R. Hahnloser. 2020. Character-level translation with self-attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1591–1604, Online. Association for Computational Linguistics.

Nathan Godey, Roman Castagné, Éric de la Clergerie, and Benoît Sagot. 2022. MANTa: Efficient gradient-based tokenization for end-to-end robust language modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2859–2870, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjan Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

Alex Graves. 2013. Generating sequences with recurrent neural networks.

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

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2741–2749. AAAI Press.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

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

Jiahuan Li, Yutong Shen, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2021. When is char better than subword: A systematic study of segmentation algorithms 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 2: Short Papers)*, pages 543–549, Online. Association for Computational Linguistics.

Jindřich Libovický and Alexander Fraser. 2020. Towards reasonably-sized character-level transformer NMT by finetuning subword systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2572–2579, Online. Association for Computational Linguistics.

Jindřich Libovický, Helmut Schmid, and Alexander Fraser. 2022. Why don't people use character-level machine translation? In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2470–2485, Dublin, Ireland. Association for Computational Linguistics.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. CharBERT: Character-aware pre-trained language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 39–50, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, Xinyi Wang, and John Wieting. 2019. compare-mt: A tool for holistic comparison of language generation systems. *CoRR*, abs/1903.07926.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880, Brussels, Belgium. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. 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 3118–3135, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Uri Shaham and Omer Levy. 2021. 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. Association for Computational Linguistics.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. pages 1017–1024.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2022. Charformer: Fast character transformers via gradient-based subword tokenization.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

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

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

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. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021. Byt5: Towards a token-free future with pre-trained byte-to-byte models.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2017. Recurrent highway networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 4189–4198. PMLR.

Appendix

A Datasets used for Multilingual Training

Language	OPUS Corpora
deu-eng	Wikipedia, WMT-News, Bible
hin-eng	IITB Hindi-English corpus
khm-eng	CCAligned, GlobalVoices, QED, GNOME, TED2020, KDE4, tico-19, Tatoeba
lao-eng	Wikimedia, TED2020, QED, GNOME, Ubuntu, Tatoeba
tel-eng	Wikimedia, TED2020, QED, GNOME, Bible
tam-eng	Wikimedia, TED2020, QED, GNOME, Tanzil
nep-eng	Wikimedia, TED2020, QED, GNOME, Bible, GlobalVoices

Table 6: List of datasets from OPUS we use to construct our corpus for multilingual experiments.

B Computational Efficiency

For comparing the byte based and subword models in terms of the number of parameters, training and inference times we consider the transformer base architecture. For the subword baseline, we consider a source vocabulary of 32k and 8k for the target vocabulary (English).

Parameters When comparing the number of parameters in subword and byte based models, we find that byte based models have far fewer parameters (~30% fewer) as compared to the subword baselines. We highlight the differences in Table 7.

	Model	#Parameters
ONE-HOT	Byte	44.1M
	Byte-nCF	46.5M
	Byte-WSF	44.1M
DENSE	Subword	68.7M
	Byte	44.3M
	Byte-nCF	46.7M
	Byte-WSF	44.1M

Table 7: Comparison of number of parameters in subword and byte based models. We find that byte based models have on average 30% fewer parameters than comparable subword models.

Training and Inference times For comparing the training times, we train the models on the WMT19 de-en data for 5k steps. We use a warmup of 1k steps to eliminate any hardware discrepancies like GPU[9] cold starts. Since the byte based models are smaller than comparable subword models, they are 20% faster to train. The inference times are based on evaluating the model on the validation set across all batches[10]. We find that byte based models are significantly slower than subword models for inference. The byte sequences are significantly longer than subword sequences and thus the decoding time takes a hit. Table 8 shows the training and inference times for the subword and byte baesd models.

It should be noted that while the training time is shorter for byte-based approaches when comparing the same number of gradient steps, when we consider the training time for the same number of epochs, we do not observe faster training performance. Since byte sequences are much longer than subword sequences, training them for the same number of epochs involves using a longer number of training steps which makes their training times comparable.

[9]All numbers are obtained using a single RTX 3090 GPU using a batch size of 7k tokens and 8 gradient accumulation steps for training.

[10]beam size of 3 and batch size of 256

	Model	Train time(s)	Inference time(s)
ONE-HOT	Byte	4184.1	50.963
	Byte-nCF	4387.6	52.315
DENSE	Subword	5176.8	22.445

Table 8: Comparison of the training time and inference time of subword and byte models. Byte-based models are faster to train, but are slower during inference than subword models.

Computing Infrastructure All models are trained on a Linux server with 4 RTX 3090 GPUs and 16 CPU cores. On average, training all models on 2 GPUs for 200K steps can be finished within 24 hours. After training, we pick the best checkpoints based on the performance on the development set.

C Number of Shallow Encoding Layers for Byte-nCF

#Layer	0	1	2	3	4	5
deu-eng	19.4	**21.5**	21.4	20.1	20.4	19.7
khm-eng	10.4	**12.6**	12.3	11.8	11.3	10.7

Table 9: BLEU score of Byte-nCF using L_s shallow encoding layers.

D Number of Word-based Self-Attention Layers for Byte-WSF

#Layer	1	2	3	4	5
deu-eng	28.3	27.9	28.8	28.2	27.4

Table 10: BLEU score of Byte-WSF using L_s word-based self-attention layers.

E Byte-BPE Baseline

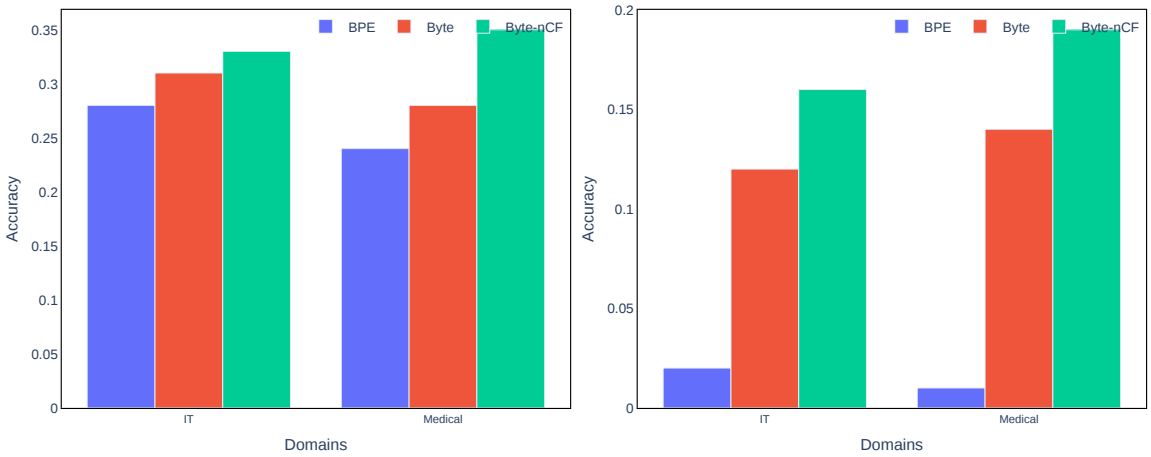
There are some works exploring the use of BPE vocabulary on byte tokens (Wang et al., 2019a) to get the best of both worlds - i.e. we would not have the out-of-vocabulary issue since every character can be represented as one of the 256 byte tokens and we also make use of the advantages of subword tokenization scheme to reduce the sequence length and decoding time. For a more reasonable comparison with similar-sized byte-based models, we strictly follow the settings of Wang et al. (2019b), using BBPE models and setting vocab size to 2K or 4K. From our results below, we find that our proposed methods for byte fusion (using 256 vocab size) are slightly better than BBPE with 4K vocab size, with an avg. gain of up to 0.8 BLEU.

Note that BBPE may fall back to using single bytes when dealing with new byte combinations in a new language or domain. We also run a cross-lingual transfer experiment by training the BBPE(4k) model on tur-eng and evaluating it on aze-eng in a zero-shot manner. We find that it gets a BLEU of 7.4, which is better than vanilla byte models (6.9) but worse than our proposed Byte-nCF (7.8 BLEU). This suggests that even though there is a byte fall-back in such models, a certain fraction of BPE tokens used in high-resource languages may not be used in the low-resource language, and the model still has to implicitly fuse the new byte tokens for low-resource language, similar to vanilla byte baseline.

This baseline is more comparable to BPE and is not the main baseline for our consideration since we are focused on improving over the vanilla byte-based methods.

Language Pairs	BBPE 2K	BBPE 4K
deu-eng	30.8	31.1
hin-eng	24.7	25.1
npi-eng	18.4	18.7
tam-eng	19.3	19.6
tel-eng	20.4	21.1
khm-eng	11.2	11.6
lao-eng	6.1	6.4
Avg.	18.7	19.1

Table 11: BLEU scores for BBPE



(a) Word accuracy for seen words during training

(b) Word accuracy on unseen words during training

Figure 4: Translation performance (indicated by Word Accuracy) grouped by whether the words were seen or unseen during training on different domains for ONE-HOT Byte models on deu-eng.

s

F Seen vs Unseen words - Domain Adaptation

To compare how well the models generalize across domains, we compute the word accuracy score based on whether the source words were observed or unobserved during the training stage. Since all the models are trained on the WMT News domain and evaluated on the IT and Medical domains, analyzing the word accuracy on unseen words reveals where the performance gain stems from. We see that the Byte-nCF model has much higher word accuracies on the unseen words when compared with the Byte and BPE models

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Left blank.

☑ A2. Did you discuss any potential risks of your work?
Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Not applicable. Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☑ Did you run computational experiments?

Left blank.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Left blank.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Left blank.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.