Neural Machine Translation without Embeddings

Uri Shaham[♢] **Omer Levy**[♢♠]

[♢]The Blavatnik School of Computer Science, Tel Aviv University
[♠]Facebook AI Research

Abstract

Many NLP models operate over sequences of subword tokens produced by hand-crafted tokenization rules and heuristic subword induction algorithms. A simple universal alternative is to represent every computerized text as a sequence of bytes via UTF-8, obviating the need for an embedding layer since there are fewer token types (256) than dimensions. Surprisingly, replacing the ubiquitous embedding layer with one-hot representations of each byte does not hurt performance; experiments on byte-to-byte machine translation from English to 10 different languages show a consistent improvement in BLEU, rivaling character-level and even standard subword-level models. A deeper investigation reveals that the combination of embeddingless models with decoder-input dropout amounts to token dropout, which benefits byte-to-byte models in particular.[1]

1 Introduction

Neural NLP models often operate on the subword level, which requires language-specific tokenizers (Koehn et al., 2007; Adler and Elhadad, 2006) and subword induction algorithms, such as BPE (Sennrich et al., 2016; Kudo, 2018). Instead, working at the byte level by representing each character as a variable number of Unicode (UTF-8) bytes, does not require any form of preprocessing, allowing the model to read and predict every computerized text using a single vocabulary of 256 types. While previous work found that byte-level models tend to underperform models based on subword tokens (Wang et al., 2019), byte-based models exhibit an interesting property: their vocabulary is smaller than the number of latent dimensions ($256 < d$). In this work, we demonstrate that this property allows us to *remove* the input and output embedding layers from byte-to-byte translation models, and in doing so, *improve* the models' performance consistently.

We replace the dense trainable embedding matrix with a fixed one-hot encoding of the vocabulary as the first and last layers of a standard transformer model. Machine translation experiments on 10 language pairs show that byte-to-byte models *without* an embedding layer achieve higher BLEU scores than byte-based models with parameterized embeddings (+0.5 on average), thus closing the performance gap with subword and character models. We observe this result consistently throughout a wide variety of target languages and writing systems.

The fact that removing parameters improves performance is counter-intuitive, especially given recent trends in machine learning that advocate for increasingly larger networks. We further investigate why embeddingless models yield better results and find implicit token dropout (commonly referred to as "word dropout") as the main source of that boost. While prior work shows that randomly masking tokens from the decoder input can improve the performance of language generation models (Bowman et al., 2016), we find that this effect is amplified when operating at the byte level. Overall, our results suggest that, even without additional parameters, byte-based models can compete and potentially outperform subword models, but that they may require alternative optimization techniques to achieve that goal.

2 Byte Tokenization

Modern software typically represents text using Unicode strings (UTF-8), which allows one to encode virtually any writing system using a variable number of bytes per token; English characters are typically represented by a single byte, with other writing systems taking two (e.g. Arabic), three (e.g. Chinese), or four (e.g. emojis) bytes per character. By treating each byte as a separate token, we can encode any natural language text using a single uni-

[1]Our code is publicly available at: https://github.com/UriSha/EmbeddinglessNMT

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 181–186
June 6–11, 2021.

Original Text	Будь здоров.																															
Subwords (BPE)	Бу@				дь					здо@						ров							.									
Characters	Б		у		д		ь			з		д		о		р		о		в			.									
Bytes (UTF-8)	D0	91	D1	83	D0	B4	D1	8C	20	D0	B7	D0	B4	D0	BE	D1	80	D0	BE	D0	B2	2E										

Figure 1: Subword (BPE), character, and byte tokens of the string "Будь здоров." UTF-8 uses two bytes to represent each character in the Cyrillic script, making the byte sequence longer than the number of characters.

versal vocabulary of only 256 token types. Moreover, byte tokenization obviates the need for any heuristic preprocessing, such as splitting spaces, punctuation, and contractions. Figure 1 illustrates subword, character, and byte tokenization.

3 Embeddingless Model

Our model is based on the original transformer encoder-decoder (Vaswani et al., 2017) with one main difference: we eliminate the input and output token embedding layers. These layers typically use a common parameter matrix $E \in R^{|V| \times d}$ that contains a d-dimensional embedding vector for each source and target vocabulary item in V.[2]

Instead, we use a fixed one-hot representation of our byte vocabulary. For instance, the character "R" could be represented as a vector with 1 at dimension 82 and 0 elsewhere. Since it is standard practice to use representations of more than 256 dimensions, every possible byte can be represented by such one-hot vectors. To predict the next token for a decoder input of n tokens, we take the output of the last transformer decoder layer, $Y \in R^{n \times d}$, and apply a softmax across each vector's dimensions. Formal expressions of the input and output of our model are detailed in Figure 2.

Omitting the embedding layer reduces the number of parameters by a factor of $O(|V| \cdot d)$.[3] We do add a total of 3 parameters to scale the encoder and decoder's (one-hot) inputs and the decoder's output (before the softmax). We initialize all three with $\sqrt{d}$, akin to the constant scaling factor typically applied to the input embedding layer in transformers. Despite the reduction in model size, memory consumption increases when working on longer sequences, since the space complexity of transformers is $O(n^2 + n \cdot d)$. In our case, d (512) is typically larger than n (see Table 1), entailing an increase in memory consumption that is roughly linear in the sequence length n, and a similar decrease in processing speed when compared to character and subword models.

	Original	**Embeddingless**
Input	$XE + P_n$	$X + P_n$
Output	$\text{softmax}_{\lvert V \rvert}\left(YE^\top\right)$	$\text{softmax}_d\left(Y\right)$

Figure 2: The main differences between the original encoder-decoder model and the new embeddingless model. $X \in R^{n \times |V|}$ is the one-hot representation of n input tokens (bytes); P_n are the positional embeddings up to length n.

In addition to replacing the embedding layers, we also remove the dropout layers on the encoder input and decoder output, since zeroing out entries of one-hot vectors is equivalent to randomly masking out input tokens or deleting significant parts of the model's predicted distribution. The dropout on the decoder input (prefix of the target fed with teacher forcing) remains intact at this point and is applied throughout our main experiments. Further analysis shows that decoder input dropout is in fact a significant source of performance gains, which we further investigate in Section 6.

4 Experiments

We train byte-tokenized embeddingless models for machine translation and compare them to standard byte, character, and subword-based models on a diverse set of languages. We adopt a standard experimental setup that was designed and tuned for the subword baseline and limits our hyperparameter tuning to dropout probabilities.

Datasets We use the IWSLT[4] datasets of English TED talks translated into other languages (Cettolo

[2] One could argue that the first layer of each transformer stack (the key, query, and value matrices) qualify as some form of multi-head multi-purpose embedding layer, where each token type is effectively represented by $3h$ different vectors (h being the number of attention heads) in the encoder and $3h$ additional vectors in the decoder. This is very different from the standard notion of embeddings, where each token type has a universal representation that can be shared across the encoder input, decoder input, and decoder output.

[3] For subword tokenization, this accounts for a significant portion of the parameter budget, but for byte-based models the added parameter cost is negligible.

[4] All languages used the IWSLT2014 data except for Vietnamese (IWSLT2015) and Japanese (IWSLT2017).

Language	ID	#Sentences	Average length		
			BPE	Char	Byte
Chinese	zh	166k	20.9	32.4	90.1
Spanish	es	167k	25.4	100.2	100.2
Arabic	ar	166k	24.4	79.3	142.2
Russian	ru	164k	26.3	93.9	169.7
German	de	159k	26.6	106.5	107.9
Japanese	ja	215k	20.9	42.4	115.3
Turkish	tr	143k	24.1	93.6	102.0
Vietnamese	vi	124k	26.9	99.8	132.5
Farsi	fa	100k	27.4	93.1	165.9
Hebrew	he	171k	23.0	72.8	129.2
English	en	-	25.6	97.0	97.1

Table 1: Languages from the IWSLT dataset, along with the number of sentence pairs in the training set and the average sequence length per tokenization method.

et al., 2014), selecting 10 additional languages with varying characteristics[5] (see Table 1). For each one, we train translation models from English to the target language (the original direction of translation), and also in the opposite direction for completeness. We clean the training data for every language pair by first removing sentences longer than 800 bytes, and then the sentences with the largest byte-length ratio between source and target such that we remove a total of 5% of the training examples.

Baselines In addition to the byte-based embeddingless transformer, we train standard transformer encoder-decoder models as baselines, each one using a different tokenization scheme: subword, character, and byte. For subword tokenization, we apply the Moses tokenizer (Koehn et al., 2007) followed by BPE (Sennrich et al., 2016). Both character and byte tokenizations apply no additional preprocessing at all and include whitespaces as valid tokens.

Hyperparameters The code for our model and baselines is based on Fairseq (Ott et al., 2019) implementation of the transformer encoder-decoder model. During preprocessing we use 10,000 merging steps when building the BPE vocabulary for every language pair. The vocabularies and embeddings are always shared among source and target languages. In every transformer we use 6 encoder and decoder layers, 4 attention heads, a hidden dimension of 512, and a feed-forward dimension of 1024. We optimize with Adam (Kingma and Ba, 2014), using the inverse square root learning rate scheduler with 4000 warmup steps and a peak learning rate of 5×10^{-4}, label smoothing of 0.1, and weight decay of 1×10^{-4}. We train each model for 50k steps and average the top 5 checkpoints according to the validation loss. We tune dropout (0.2 or 0.3) on the validation set. We set the batch size according to a maximum of 64,000 bytes per batch, which controls for the number of batches per epoch across different tokenization methods.

[5]While in this work we prioritized language and writing system diversity, there is room to test embedingless models on larger datasets in future work.

Benchmark		Embedding-based Models			Embed-less
Src	Tgt	Subword	Char	Byte	Byte
en	zh	19.9	20.8	20.2	**21.0**
en	es	**36.8**	36.3	36.3	**36.8**
en	ar	12.5	12.5	12.3	**12.9**
en	ru	18.1	17.6	17.4	**18.2**
en	de	**29.4**	28.6	28.7	29.1
en	ja	12.0	12.5	12.5	**13.1**
en	tr	13.6	13.7	13.8	**14.1**
en	vi	**29.7**	28.2	28.0	28.7
en	fa	11.5	11.7	12.0	**12.1**
en	he	26.1	**26.9**	26.4	26.7
zh	en	**16.8**	16.6	15.6	16.1
es	en	**39.6**	38.5	38.4	38.8
ar	en	**31.5**	30.2	30.3	30.8
ru	en	**22.7**	21.9	22.0	22.0
de	en	**35.4**	34.0	34.1	34.5
ja	en	**13.1**	12.6	11.4	12.2
tr	en	**23.3**	22.5	22.3	**23.3**
vi	en	**26.8**	25.0	24.7	25.3
fa	en	**23.5**	22.4	22.1	22.6
he	en	**37.8**	36.9	37.0	37.4

Table 2: Test BLEU scores of the baseline and embeddingless models on the IWSLT dataset.

Evaluation We evaluate our models using SacreBLEU, case-sensitive, with the 13a tokenizer for all languages except Chinese (ZH tokenizer) and Japanese (MeCab tokenizer). We use the raw text as the reference for all of our experiments, instead of using the default tokenized-detokenized version, which normalizes the text and gives an artificial advantage to text processed with Moses.

5 Results

Table 2 shows our experiments' results. Every row describes the test BLEU scores of our model and the three baselines trained on a different language pair. We discuss the implications of these below.

Are embeddings essential? The results show that it is indeed possible to train embeddingless machine translation models that perform competitively. The performance gaps between models with different tokenization schemes are relatively small. Except for Vietnamese, the difference between

the embeddingless model and the best embedding-based model is always under 1 BLEU.

In the most controlled setting, where we compare byte-based models with and without learnable embeddings, models *without* embeddings consistently achieve higher BLEU scores in 19 of 20 cases (and an equal score for ru-en), with a boost of about 0.5 BLEU on average. When compared to models based on character embeddings, the embeddingless byte-to-byte approach yields higher BLEU scores in 17 out of 20 cases, though the average difference is quite small in practice (0.3 BLEU).

Is subword tokenization superior to bytes or characters? Previous work in machine translation shows that subword models consistently outperform character or byte-based models (Gupta et al., 2019; Wang et al., 2019; Gao et al., 2020). However, our results indicate that this is not necessarily the case. When translating from English to a foreign language, the *original* direction of the IWSLT dataset, embeddingless byte-to-byte models achieve performance that is equal or better than subword embedding models' in 8 out of 10 cases. We observe a different trend when translating into English, where subword models surpass other models for every source language; the fact that Moses is a particularly good tokenizer for English – and less so for other languages – is perhaps related to this phenomenon. Whereas prior work proposed closing the performance gap by adding layers to the basic architecture, under the assumption that character-based models lack capacity or expressiveness, our results show that actually *removing* a component from the model can improve performance under certain conditions. It is possible that character and byte-based transformer models encounter an optimization issue rather than one of capacity or expressivity.

6 Analysis

Why does *removing* the embedding matrix *improve* the performance of byte-based models? As mentioned in Section 3, the embeddingless models do not use dropout on the encoder input and decoder output, but do apply dropout on the decoder input while training. Since the embeddingless decoder's inputs are fixed one-hot vectors, using dropout implicitly drops out complete tokens. In prior work, token dropout ("word dropout") has been shown to have a consistently positive effect (Bowman et al., 2016). We, therefore, rerun our experiments while controlling for token dropout ($p = 0.2$) to determine its effect on our results.

	Embedding-based Models			Embed-less
	Subword	Char	Byte	Byte
en→xx	+0.33	+0.53	+0.42	+0.62
xx→en	+0.69	+0.67	+0.92	+0.83

Table 3: The validation set performance gain of token dropout (0.2), averaged across languages and model dropout values.

Table 3 shows that decoder-side token dropout improves the performance of all models, with a larger impact on byte-based models and embeddingless models in particular. This effect is largely consistent, with only 7 out of 160 cases in which token dropout decreased performance on the validation set. We suspect that dropping out target tokens softens the effects of exposure bias by injecting noise into the ground-truth prefix.

Given the benefits of token dropout on the baseline models, we re-evaluate the results from Section 5, while allowing for token dropout as a potential hyperparameter. Table 4 shows that, when translating from the original English text to a foreign language, the different models perform roughly on par, with no single tokenization method dominating the others. Furthermore, byte-level models with and without embeddings achieve almost identical results. In contrast, when translating in the opposite direction, subword models consistently outperform the other methods with an average gap of 0.76 BLEU from the next best model. Also, removing the embeddings from byte-based models decreases performance by an average of 0.45 BLEU when generating English. This discrepancy might stem from artifacts of reverse translation, or perhaps from the English-centric nature of subword tokenization, which is based on Moses preprocessing and BPE. Overall, these results suggest that despite the greater number of parameters in subword models, character and byte models can perform competitively, but may require slightly different optimization techniques to do so.

7 Related Work

There is prior work on replacing language-specific tokenizers with more universal tokenization approaches. Schütze (2017) shows how character n-gram embeddings can be effectively trained by segmenting text using a stochastic process. Sen-

Benchmark		Embedding-based Models			Embed-less
Src	Tgt	Subword	Char	Byte	Byte
en	zh	20.3	**21.2**	20.8	21.0
en	es	36.7	**36.8**	**36.8**	**36.8**
en	ar	12.7	**13.1**	12.7	12.9
en	ru	**18.5**	18.2	17.7	18.2
en	de	**29.8**	29.3	29.2	29.1
en	ja	12.4	**13.1**	12.5	**13.1**
en	tr	13.9	14.3	**14.4**	14.1
en	vi	**30.0**	29.1	28.9	28.7
en	fa	11.5	**12.2**	12.1	12.1
en	he	26.8	**27.1**	**27.1**	26.7
zh	en	**17.3**	17.2	16.3	16.1
es	en	**40.0**	39.1	39.1	38.8
ar	en	**32.0**	31.1	31.2	30.8
ru	en	**22.9**	22.4	22.5	22.0
de	en	**35.6**	34.9	35.0	34.5
ja	en	**13.5**	12.8	12.3	11.2
tr	en	**24.3**	23.3	23.7	23.3
vi	en	**27.4**	25.9	25.9	25.3
fa	en	**24.5**	23.2	23.3	22.6
he	en	**38.2**	37.8	37.4	37.4

Table 4: Test BLEU scores of the baseline and embeddingless models on the IWSLT dataset, when decoder-side token dropout is considered as a potential hyperparameter setting.

tencePiece (Kudo and Richardson, 2018) tokenizes raw Unicode strings into subwords using BPE (Sennrich et al., 2016) or unigram LM (Kudo, 2018). Byte BPE (Wang et al., 2019) extends SentencePiece to operate at the byte level. While this approach is indeed more language-agnostic than heuristic tokenizers, it does suffer from performance degradation when no pre-tokenization (e.g. splitting by whitespace) is applied.[6] Moreover, the assumption that subword units must be contiguous segments does not hold for languages with non-concatenative morphology such as Arabic and Hebrew.

Character and byte-based language models (Lee et al., 2017; Al-Rfou et al., 2019) treat the raw text as a sequence of tokens (characters or bytes) and do not require any form of preprocessing or word tokenization, and Choe et al. (2019) even demonstrate that byte-based language models can perform comparably to word-based language models on the billion-word benchmark (Chelba et al., 2013). Although earlier results on LSTM-based machine translation models show that character tokenization can outperform subword tokenization (Cherry et al., 2018), recent literature shows that the same does not hold for transformers (Gupta et al., 2019; Wang et al., 2019; Gao et al., 2020). To narrow the gap, recent work suggests using deeper models (Gupta et al., 2019) or specialized architectures (Gao et al., 2020). Our work deviates from this trend by *removing* layers to improve the model. This observation contests the leading hypothesis in existing literature – that the performance gap results from reduced model capacity – and suggests that the problem may be one of optimization.

[6] https://github.com/google/sentencepiece/blob/master/doc/experiments.md

8 Conclusions

This work challenges two key assumptions in neural machine translation models: the necessity of embedding layers, and the superiority of subword tokenization. Experiments on 10 different languages show that, despite their ubiquitous usage, competitive models can be trained without any embeddings by treating text as a sequence of bytes. Our investigation suggests that different tokenization methods may require revisiting the standard optimization techniques used with transformers, which are primarily geared towards sequences of English subwords.

Acknowledgements

This work was supported in part by Len Blavatnik and the Blavatnik Family foundation, the Alon Scholarship, and the Tel Aviv University Data Science Center.

References

Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based HMM for Hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 665–672, Sydney, Australia. 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*, 33:3159–3166.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, volume 57.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling.

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.

Dokook Choe, Rami Al-Rfou, Mandy Guo, Heeyoung Lee, and Noah Constant. 2019. Bridging the gap for tokenizer-free language models.

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.

Rohit Gupta, Laurent Besacier, Marc Dymetman, and Matthias Gallé. 2019. Character-based nmt with transformer. *arXiv preprint arXiv:1911.04997*.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

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.

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

Hinrich Schütze. 2017. Nonsymbolic text representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 785–796, Valencia, Spain. Association for Computational Linguistics.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

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