BPE-Dropout: Simple and Effective Subword Regularization

Ivan Provilkov[∗ 1,2] **Dmitrii Emelianenko**[∗ 1,3] **Elena Voita**[4,5]

[1]Yandex, Russia
[2]Moscow Institute of Physics and Technology, Russia
[3]National Research University Higher School of Economics, Russia
[4]University of Edinburgh, Scotland [5]University of Amsterdam, Netherlands
{iv-provilkov, dimdi-y, lena-voita}@yandex-team.ru

Abstract

Subword segmentation is widely used to address the open vocabulary problem in machine translation. The dominant approach to subword segmentation is Byte Pair Encoding (BPE), which keeps the most frequent words intact while splitting the rare ones into multiple tokens. While multiple segmentations are possible even with the same vocabulary, BPE splits words into unique sequences; this may prevent a model from better learning the compositionality of words and being robust to segmentation errors. So far, the only way to overcome this BPE imperfection, its deterministic nature, was to create another subword segmentation algorithm (Kudo, 2018). In contrast, we show that BPE itself incorporates the ability to produce multiple segmentations of the same word. We introduce *BPE-dropout* – simple and effective subword regularization method based on and compatible with conventional BPE. It stochastically corrupts the segmentation procedure of BPE, which leads to producing multiple segmentations within the same fixed BPE framework. Using *BPE-dropout* during training and the standard BPE during inference improves translation quality up to 2.3 BLEU compared to BPE and up to 0.9 BLEU compared to the previous subword regularization.

1 Introduction

Using subword segmentation has become de-facto standard in Neural Machine Translation (Bojar et al., 2018; Barrault et al., 2019). Byte Pair Encoding (BPE) (Sennrich et al., 2016) is the dominant approach to subword segmentation. It keeps the common words intact while splitting the rare and unknown ones into a sequence of subword units. This potentially allows a model to make use of morphology, word composition and transliteration. BPE effectively deals with an open-vocabulary problem and is widely used due to its simplicity.

There is, however, a drawback of BPE in its deterministic nature: it splits words into unique subword sequences, which means that for each word a model observes only one segmentation. Thus, a model is likely not to reach its full potential in exploiting morphology, learning the compositionality of words and being robust to segmentation errors. Moreover, as we will show further, subwords into which rare words are segmented end up poorly understood.

A natural way to handle this problem is to enable multiple segmentation candidates. This was initially proposed by Kudo (2018) as a subword regularization – a regularization method, which is implemented as an on-the-fly data sampling and is not specific to NMT architecture. Since standard BPE produces single segmentation, to realize this regularization the author had to propose a new subword segmentation, different from BPE. However, the introduced approach is rather complicated: it requires training a separate segmentation unigram language model, using EM and Viterbi algorithms, and forbids using conventional BPE.

In contrast, we show that BPE itself incorporates the ability to produce multiple segmentations of the same word. BPE builds a vocabulary of subwords and a merge table, which specifies which subwords have to be merged into a bigger subword, as well as the priority of the merges. During segmentation, words are first split into sequences of characters, then the learned merge operations are applied to merge the characters into larger, known symbols, till no merge can be done (Figure 1(a)). We introduce *BPE-dropout* – a subword regularization method based on and compatible with conventional BPE. It uses a vocabulary and a

[∗]Equal contribution.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1882–1892
July 5 - 10, 2020.

```
u-n-r-e-l-a-t-e-d
u-n re-l-a-t-e-d
u-n re-l-at-e-d
u-n re-l-at-ed
un re-l-at-ed
un re-l-ated
un rel-ated
un-related
unrelated
```

(a)

```
u-n_r-e-l-a_t-e_d
u-n re-l_a-t-e_d
u-n re_l-at-e_d
un re-l-at-e-d
un re_l-at-ed
un re-lat-ed
un relat_ed
```

```
u-n-r-e-l-a_t-e-d
u_n re_l-a-t-e-d
u_n re-l-at-e-d
u_n re-l-ate_d
u_n rel-ate-d
u_n relate_d
```

```
u-n_r_e_l-a-t-e-d
u-n-r_e-l-at-e-d
u-n-r_e-l_at_ed
un-r-e-l-at-ed
un re-l_at-ed
un re-l-ated
un rel_ated
```

(b)

Figure 1: Segmentation process of the word '*unrelated*' using (a) BPE, (b) *BPE-dropout*. Hyphens indicate possible merges (merges which are present in the merge table); merges performed at each iteration are shown in green, dropped – in red.

merge table built by BPE, but at each merge step, some merges are randomly dropped. This results in different segmentations for the same word (Figure 1(b)). Our method requires no segmentation training in addition to BPE and uses standard BPE at test time, therefore is simple. *BPE-dropout* is superior compared to both BPE and Kudo (2018) on a wide range of translation tasks, therefore is effective.

Our key contributions are as follows:

- We introduce *BPE-dropout* – a simple and effective subword regularization method;
- We show that our method outperforms both BPE and previous subword regularization on a wide range of translation tasks;
- We analyze how training with *BPE-dropout* affects a model and show that it leads to a better quality of learned token embeddings and to a model being more robust to noisy input.

2 Background

In this section, we briefly describe BPE and the concept of subword regularization. We assume that our task is machine translation, where a model needs to predict the target sentence Y given the source sentence X, but the methods we describe are not task-specific.

2.1 Byte Pair Encoding (BPE)

To define a segmentation procedure, BPE (Sennrich et al., 2016) builds a token vocabulary and a merge table. The token vocabulary is initialized with the character vocabulary, and the merge table is initialized with an empty table. First, each word is represented as a sequence of tokens plus a special end of word symbol. Then, the method iteratively counts all pairs of tokens and merges the most frequent pair into a new token. This token is added to the vocabulary, and the merge operation is added to the merge table. This is done until the desired vocabulary size is reached.

The resulting merge table specifies which subwords have to be merged into a bigger subword, as well as the priority of the merges. In this way, it defines the segmentation procedure. First, a word is split into distinct characters plus the end of word symbol. Then, the pair of adjacent tokens which has the highest priority is merged. This is done iteratively until no merge from the table is available (Figure 1(a)).

2.2 Subword regularization

Subword regularization (Kudo, 2018) is a training algorithm which integrates multiple segmentation candidates. Instead of maximizing log-likelihood, this algorithm maximizes log-likelihood marginalized over different segmentation candidates. Formally,

$$\mathcal{L} = \sum_{(X,Y)\in D} \mathop{\mathbb{E}}_{\substack{x\sim P(x|X)\\ y\sim P(y|Y)}} \log P(y|x,\theta), \qquad (1)$$

where x and y are sampled segmentation candidates for sentences X and Y respectively, $P(x|X)$ and $P(y|Y)$ are the probability distributions the candidates are sampled from, and θ is the set of model parameters. In practice, at each training step only one segmentation candidate is sampled.

Since standard BPE segmentation is deterministic, to realize this regularization Kudo (2018) proposed a new subword segmentation. The introduced approach requires training a separate segmentation unigram language model to predict the probability of each subword, EM algorithm to optimize the vocabulary, and Viterbi algorithm to make samples of segmentations.

Subword regularization was shown to achieve significant improvements over the method using a single subword sequence. However, the proposed method is rather complicated and forbids using conventional BPE. This may prevent practitioners from using subword regularization.

3 Our Approach: BPE-Dropout

We show that to realize subword regularization it is not necessary to reject BPE since multiple segmentation candidates can be generated within the BPE framework. We introduce *BPE-dropout* – a method which exploits the innate ability of BPE to be stochastic. It alters the segmentation procedure while keeping the original BPE merge table. During segmentation, at each merge step some merges are randomly dropped with the probability p. This procedure is described in Algorithm 1.

Algorithm 1: BPE-dropout

```
current_split ← characters from input_word;
do
    merges ← all possible merges[1] of tokens
      from current_split;
    for merge from merges do
        /* The only difference
           from BPE                    */
        remove merge from merges with the
          probability p;
    end
    if merges is not empty then
        merge ← select the merge with the
          highest priority from merges;
        apply merge to current_split;
    end
while merges is not empty;
return current_split;
```

If p is set to 0, the segmentation is equivalent to the standard BPE; if p is set to 1, the segmentation splits words into distinct characters. The values between 0 and 1 can be used to control the segmentation granularity.

We use $p > 0$ (usually $p = 0.1$) in training time to expose a model to different segmentations and $p = 0$ during inference, which means that at inference time we use the original BPE. We discuss the choice of the value of p in Section 5.

When some merges are randomly forbidden during segmentation, words end up segmented in different subwords; see for example Figure 1(b). We hypothesize that exposing a model to different segmentations may result in better understanding of the whole words as well as their subword units; we will verify this in Section 6.

4 Experimental setup

4.1 Baselines

Our baselines are the standard BPE and the subword regularization by Kudo (2018).

Subword regularization by Kudo (2018) has segmentation sampling hyperparameters l and α. l specifies how many best segmentations for each word are produced before sampling one of them, α controls the smoothness of the sampling distribution. In the original paper ($l = \infty, \alpha = 0.2/0.5$) and ($l = 64, \alpha = 0.1$) were shown to perform best on different datasets. Since overall they show comparable results, in all experiments we use ($l = 64, \alpha = 0.1$).

4.2 Vocabularies

There are two ways of building vocabulary for models trained with *BPE-dropout*: (1) take the vocabulary built by BPE; then the segmented with *BPE-dropout* text will contain a small number of unknown tokens (UNKs)[2]; (2) add to the BPE vocabulary all tokens which can appear when segmenting with *BPE-dropout*.

In the preliminary experiments, we did not observe any difference in quality; therefore, either of the methods can be used. We choose the first option to stay in the same setting as the standard BPE. Besides, a model exposed to some UNKs in training can be more reliable for practical applications where unknown tokens can be present.

4.3 Data sets and preprocessing

We conduct our experiments on a wide range of datasets with different corpora sizes and languages; information about the datasets is summarized in Table 1. These datasets are used in the main experiments (Section 5.1) and were chosen to match the ones used in the prior work (Kudo, 2018). In the additional experiments (Sections 5.2-5.5), we also use random subsets of the WMT14 English-French data; in this case, we specify dataset size for each experiment.

Prior to segmentation, we preprocess all

[1] In case of multiple occurrences of the same merge in a word (for example, `m-e-r-g-e-r` has two occurrences of the merge `(e, r)`), we decide independently for each occurrence whether to drop it or not.

[2] For example, for the English part of the IWSLT15 En-Vi corpora, these UNKs make up 0.00585 and 0.00085 of all tokens for 32k and 4k vocabularies, respectively.

		Number of sentences (train/dev/test)	Voc size	Batch size	The value of p in *BPE-dropout*
IWSLT15	En ↔ Vi	133k / 1553 / 1268	4k	4k	0.1 / 0.1
	En ↔ Zh	209k / 887 / 1261	4k / 16k	4k	0.1 / 0.6
IWSLT17	En ↔ Fr	232k / 890 / 1210	4k	4k	0.1 / 0.1
	En ↔ Ar	231k / 888 / 1205	4k	4k	0.1 / 0.1
WMT14	En ↔ De	4.5M / 3000 / 3003	32k	32k	0.1 / 0.1
ASPEC	En ↔ Ja	2M / 1700 / 1812	16k	32k	0.1 / 0.6

Table 1: Overview of the datasets and dataset-dependent hyperparametes; values of p are shown in pairs: source language / target language. (We explain the choice of the value of p for *BPE-dropout* in Section 5.3.)

datasets with the standard Moses toolkit.[3] However, Chinese and Japanese have no explicit word boundaries, and Moses tokenizer does not segment sentences into words; for these languages, subword segmentations are trained almost from unsegmented raw sentences.

Relying on a recent study of how the choice of vocabulary size influences translation quality (Ding et al., 2019), we choose vocabulary size depending on the dataset size (Table 1).

In training, translation pairs were batched together by approximate sequence length. For the main experiments, the values of batch size we used are given in Table 1 (batch size is the number of source tokens). In the experiments in Sections 5.2, 5.3 and 5.4, for datasets not larger than 500k sentence pairs we use vocabulary size and batch size of 4k, and 32k for the rest.[4]

In the main text, we train all models on lowercased data. In the appendix, we provide additional experiments with the original case and case-sensitive BLEU.

4.4 Model and optimizer

The NMT system used in our experiments is *Transformer base* (Vaswani et al., 2017). More precisely, the number of layers is $N = 6$ with $h = 8$ parallel attention layers, or heads. The dimensionality of input and output is $d_{model} = 512$, and the inner-layer of feed-forward networks has dimensionality $d_{ff} = 2048$. We use regularization and optimization procedure as described in Vaswani et al. (2017).

4.5 Training time

We train models till convergence. For all experiments, we provide number of training batches in the appendix (Tables 6 and 7).

4.6 Inference

To produce translations, for all models, we use beam search with the beam of 4 and length normalization of 0.6.

In addition to the main results, Kudo (2018) also report scores using n-best decoding. To translate a sentence, this strategy produces multiple segmentations of a source sentence, generates a translation for each of them, and rescores the obtained translations. While this could be an interesting future work to investigate different sampling and rescoring strategies, in the current study we use 1-best decoding to fit in the standard decoding paradigm.

4.7 Evaluation

For evaluation, we average 5 latest checkpoints and use BLEU (Papineni et al., 2002) computed via SacreBleu[5] (Post, 2018). For Chinese, we add option `--tok zh` to SacreBLEU. For Japanese, we use character-based BLEU.

5 Experiments

5.1 Main results

The results are provided in Table 2. For all datasets, *BPE-dropout* improves significantly over the standard BPE: more than 1.5 BLEU for En-Vi, Vi-En, En-Zh, Zh-En, Ar-En, De-En, and 0.5-1.4

[3] `https://github.com/moses-smt/mosesdecoder`

[4] Large batch size can be reached by using several of GPUs or by accumulating the gradients for several batches and then making an update.

[5] Our SacreBLEU signature is: `BLEU+case.lc+lang.[src-lang]-[dst-lang]+numrefs.1+smooth.exp+tok.13a+version.1.3.6`

	BPE	Kudo (2018)	*BPE-dropout*
IWSLT15			
En-Vi	31.78	32.43	**33.27**
Vi-En	30.83	32.36	**32.99**
En-Zh	20.48	**23.01**	**22.84**
Zh-En	19.72	21.10	**21.45**
IWSLT17			
En-Fr	39.37	39.45	**40.02**
Fr-En	38.18	38.88	**39.39**
En-Ar	13.89	14.43	**15.05**
Ar-En	31.90	32.80	**33.72**
WMT14			
En-De	27.41	**27.82**	**28.01**
De-En	32.69	33.65	**34.19**
ASPEC			
En-Ja	54.51	**55.46**	55.00
Ja-En	30.77	**31.23**	**31.29**

Table 2: BLEU scores. Bold indicates the best score and all scores whose difference from the best is not statistically significant (with p-value of 0.05). (Statistical significance is computed via bootstrapping (Koehn, 2004).)

BLEU for the rest. The improvements are especially prominent for smaller datasets; we will discuss this further in Section 5.4.

Compared to Kudo (2018), among the 12 datasets we use *BPE-dropout* is beneficial for 8 datasets with improvements up to 0.92 BLEU, is not significantly different for 3 datasets and underperforms only on En-Ja. While Kudo (2018) uses another segmentation, our method operates within the BPE framework and changes only the way a model is trained. Thus, lower performance of *BPE-dropout* on En-Ja and only small or insignificant differences for Ja-En, En-Zh and Zh-En suggest that Japanese and Chinese may benefit from a language-specific segmentation.

Note also that Kudo (2018) report larger improvements over BPE from using their method than we show in Table 2. This might be explained by the fact that Kudo (2018) used large vocabulary size (16k, 32k), which has been shown counterproductive for small datasets (Sennrich and Zhang, 2019; Ding et al., 2019). While this may not be the issue for models trained with subword regularization (see Section 5.4), this causes drastic drop in performance of the baselines.

	BPE	*BPE-dropout*		
		src-only	dst-only	both
250k	26.94	27.98	27.71	**28.40**
500k	29.28	**30.12**	29.40	**29.89**
1m	30.53	**31.09**	30.62	**31.23**
4m	33.38	**33.89**	33.46	**33.85**
16m	34.37	**34.82**	-	33.66

Table 3: BLEU scores for models trained with *BPE-dropout* on a single side of a translation pair or on both sides. Models trained on random subsets of WMT14 En-Fr dataset. Bold indicates the best score and all scores whose difference from the best is not statistically significant (with p-value of 0.05).

5.2 Single side vs full regularization

In this section, we investigate whether *BPE-dropout* should be used only on one side of a translation pair or for both source and target languages. We select random subsets of different sizes from WMT14 En-Fr data to understand how the results are affected by the amount of data. We show that:

- for small and medium datasets, full regularization performs best;
- for large datasets, *BPE-dropout* should be used only on the source side.

Since full regularization performs the best for most of the considered dataset sizes, in the subsequent sections we use *BPE-dropout* on both source and target sides.

5.2.1 Small and medium datasets: use full regularization

Table 3 indicates that using *BPE-dropout* on the source side is more beneficial than on the target side; for the datasets not smaller than 0.5m sentence pairs, *BPE-dropout* can be used only the source side. We can speculate that it is more important for the model to understand a source sentence than being exposed to different ways to generate the same target sentence.

5.2.2 Large datasets: use only for source

For larger corpora (e.g., starting from 4m instances), it is better to use *BPE-dropout* only on the source side (Table 3). Interestingly, using *BPE-dropout* for both source and target languages hurts performance for large datasets.

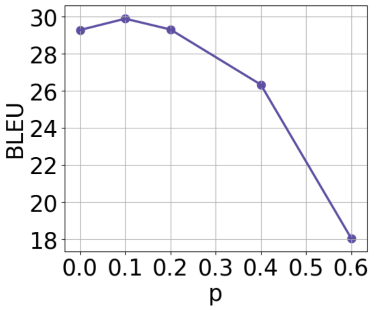


Figure 2: BLEU scores for the models trained with *BPE-dropout* with different values of p. WMT14 En-Fr, 500k sentence pairs.

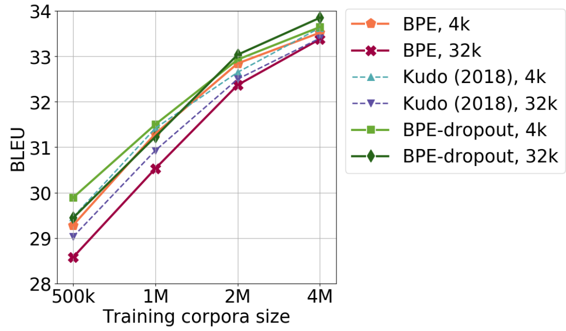


Figure 3: BLEU scores. Models trained on random subsets of WMT14 En-Fr.

5.3 Choice of the value of p

Figure 2 shows BLEU scores for the models trained on *BPE-dropout* with different values of p (the probability of a merge being dropped). Models trained with high values of p are unable to translate due to a large mismatch between training segmentation (which is close to char-level) and inference segmentation (BPE). The best quality is achieved with $p = 0.1$.

In our experiments, we use $p = 0.1$ for all languages except for Chinese and Japanese. For Chinese and Japanese, we take the value of $p = 0.6$ to match the increase in length of segmented sentences for other languages.[6]

5.4 Varying corpora and vocabulary size

Now we will look more closely at how the improvement from using *BPE-dropout* depends on corpora and vocabulary size.

First, we see that *BPE-dropout* performs best for all dataset sizes (Figure 3). Next, models trained with subword regularization are less sensitive to the choice of vocabulary size: differences in performance of models with 4k and 32k vocabulary are much less than for models trained with the standard BPE. This makes *BPE-dropout* attractive since it allows (i) not to tune vocabulary size for each dataset, (ii) choose vocabulary size depending on the desired model properties: models with smaller vocabularies are beneficial in terms of number of parameters, models with larger vocabularies are beneficial in terms of inference time.[7] Finally, we see that the effect from using *BPE-dropout* vanishes when a corpora size gets bigger. This is not surprising: the effect of any regularization is less in high-resource settings; however, as we will show later in Section 6.3, when applied to noisy source, models trained with *BPE-dropout* show substantial improvements up to 2 BLEU even in high-resource settings.

Note that for larger corpora, we recommend using *BPE-dropout* only for source language (Section 5.2).

5.5 Inference time and length of generated sequences

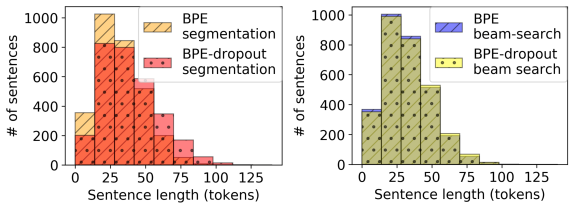
Since *BPE-dropout* produces more fine-grained segmentation, sentences segmented with *BPE-dropout* are longer; distribution of sentence lengths are shown in Figure 4 (a) (with $p = 0.1$, on average about 1.25 times longer). Thus there is a potential danger that models trained with *BPE-dropout* may tend to use more fine-grained segmentation in inference and hence to slow inference down. However, in practice this is not the case: distributions of lengths of generated translations for models trained with BPE and with *BPE-dropout* are close (Figure 4 (b)).[8]

Table 4 confirms these observations and shows that inference time of models trained with *BPE-dropout* is not substantially different from the ones trained with BPE.

[6]Formally, for English/French/etc. with *BPE-dropout*, $p = 0.1$ sentences become on average about 1.25 times longer compared to segmented with BPE; for Chinese and Japanese, we need to set the value of p to 0.6 to achieve the same increase.

[7]Table 4 shows that inference for models with 4k vocabulary is more than 1.4 times longer than models with 32k vocabulary.

[8]This is the result of using beam search: while samples from a model reproduce training data distribution quite well, beam search favors more frequent tokens (Ott et al., 2018). Therefore, beam search translations tend not to use less frequent fine-grained segmentation.



(a) (b)

Figure 4: Distributions of length (in tokens) of (a) the French part of WMT14 En-Fr test set segmented using BPE or *BPE-dropout*; and (b) the generated translations for the same test set by models trained with BPE or *BPE-dropout*.

voc size	BPE	*BPE-dropout*
32k	1.0	1.03
4k	1.44	1.46

Table 4: Relative inference time of models trained with different subword segmentation methods. Results obtained by (1) computing averaged over 1000 runs time needed to translate WMT14 En-Fr test set, (2) dividing all results by the smallest of the obtained times.

6 Analysis

In this section, we analyze qualitative differences between models trained with BPE and *BPE-dropout*. We find, that

- when using BPE, frequent sequences of characters rarely appear in a segmented text as individual tokens rather than being a part bigger ones; *BPE-dropout* alleviates this issue;
- by analyzing the learned embedding spaces, we show that using *BPE-dropout* leads to a better understanding of rare tokens;
- as a consequence of the above, models trained with *BPE-dropout* are more robust to misspelled input.

6.1 Substring frequency

Here we highlight one of the drawbacks of BPE's deterministic nature: since it splits words into unique subword sequences, only rare words are split into subwords. This forces frequent sequences of characters to mostly appear in a segmented text as part of bigger tokens, and not as individual tokens. To show this, for each token in the BPE vocabulary we calculate how often it appears in a segmented text as an individual token and as a sequence of characters (which may be part of a bigger token or an individual token). Figure 5 shows distribution of the ratio between substring frequency as an individual token and as a sequence of characters (for top-10% most frequent substrings).

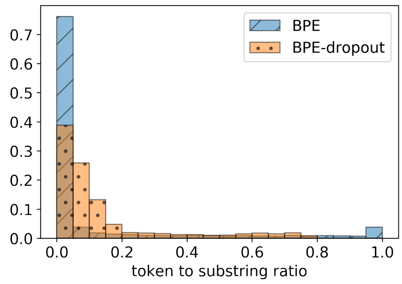


Figure 5: Distribution of token to substring ratio for texts segmented using BPE or *BPE-dropout* for the same vocabulary of 32k tokens; only 10% most frequent substrings are shown. (Token to substring ratio of a token is the ratio between its frequency as an individual token and as a sequence of characters.)

For frequent substrings, the distribution of token to substring ratio is clearly shifted to zero, which confirms our hypothesis: frequent sequences of characters rarely appear in a segmented text as individual tokens. When a text is segmented using *BPE-dropout* with the same vocabulary, this distribution significantly shifts away from zero, meaning that frequent substrings appear in a segmented text as individual tokens more often.

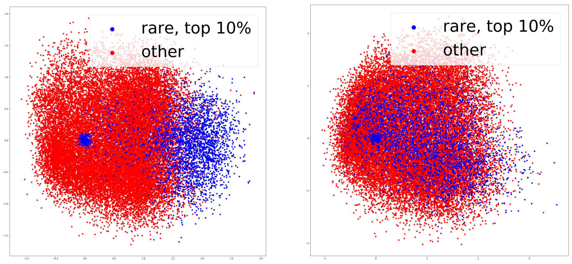
6.2 Properties of the learned embeddings

Now we will analyze embedding spaces learned by different models. We take embeddings learned by models trained with BPE and *BPE-dropout* and for each token look at the closest neighbors in the corresponding embedding space. Figure 6 shows several examples. In contrast to BPE, nearest neighbours of a token in the embedding space of *BPE-dropout* are often tokens that share sequences of characters with the original token. To verify this observation quantitatively, we computed character 4-gram precision of top-10 neighbors: the proportion of those 4-grams of the top-10 closest neighbors which are present among 4-grams of the original token. As expected, embeddings of *BPE-dropout* have higher character 4-gram precision (0.29) compared to the precision of BPE (0.18).

This also relates to the study by Gong et al. (2018). For several tasks, they analyze the em-

withdra		resul		meeting		olec		comptroll	
BPE	BPE-dropout	BPE	BPE-dropout	BPE	BPE-dropout	BPE	BPE-dropout	BPE	BPE-dropout
aimed	withd	undert	result	meetings	meetings	olecular	molec	icial	comptrollership
molecules	withdrawal	checkl	results	meet	meet	molecules	olecular	supervis	comptroller
aromatic	withdraw	maastr	resulting	session	eting	ljubl	molecule	&	troll
specialties	withdrawn	&	resulted	conference	me	zona	molecular	subcomm	controll
publishers	withdrew	unisp	ults	met	etings	choler	molecules	yugosl	contoller
chain	withdrawals	phili	res	workshop	met	oler	aec	trigg	controlled
americ	withdrawing	ς	resultant	meets	meets	ospheric	oler	sophistic	controllers
chron	dra	preca	ult	sessions	session	olar	tolu	obstac	control
eager	retire	prosecut	ul	convened	et	elic	omet	reag	contro
ighty	reti	tali	outcome	reunion	conference	ochlor	olip	entals	controls

Figure 6: Examples of nearest neighbours in the source embedding space of models trained with BPE and *BPE-dropout*. Models trained on WMT14 En-Fr (4m).



(a) BPE (b) *BPE-dropout*

Figure 7: Visualization of source embeddings. Models trained on WMT14 En-Fr (4m).

source	BPE	*BPE-dropout*	diff
En-De			
original	27.41	**28.01**	+0.6
misspelled	24.45	**26.03**	+1.58
De-En			
original	32.69	**34.19**	+1.5
misspelled	29.71	**32.03**	+2.32
En-Fr (4m)			
original	33.38	**33.85**	+0.47
misspelled	30.30	**32.13**	+1.83
En-Fr (16m)			
original	34.37	**34.82**	+0.45
misspelled	31.23	**32.94**	+1.71

Table 5: BLEU scores for models trained on WMT14 dataset evaluated given the original and misspelled source. For En-Fr trained on 16m sentence pairs, *BPE-dropout* was used only on the source side (Section 5.2).

bedding space learned by a model. The authors find that while a popular token usually has semantically related neighbors, a rare word usually does not: a vast majority of closest neighbors of rare words are rare words. To confirm this, we reduce dimensionality of embeddings by SVD and visualize (Figure 7). For the model trained with BPE, rare tokens are in general separated from the rest; for the model trained with *BPE-dropout*, this is not the case. While to alleviate this issue Gong et al. (2018) propose to use adversarial training for embedding layers, we showed that a trained with *BPE-dropout* model does not have this problem.

6.3 Robustness to misspelled input

Models trained with *BPE-dropout* better learn compositionality of words and the meaning of subwords, which suggests that these models have to be more robust to noise. We verify this by measuring the translation quality of models on a test set augmented with synthetic misspellings. We augment the source side of a test set by modifying each word with the probability of 10% by applying one of the predefined operations. The operations we consider are (1) removal of one character from a word, (2) insertion of a random character into a word, (3) substitution of a character in a word with a random one. This augmentation produces words with the edit distance of 1 from the unmodified words. Edit distance is commonly used to model misspellings (Brill and Moore, 2000; Ahmad and Kondrak, 2005; Pinter et al., 2017).

Table 5 shows the translation quality of the models trained on WMT 14 dataset when given the original source and augmented with misspellings. We deliberately chose large datasets, where improvements from using *BPE-dropout* are smaller. We can see that while for the original test sets the improvements from using *BPE-dropout* are usually modest, for misspelled test set the improvements are a lot larger: 1.6-2.3 BLEU. This is especially interesting since models have not been exposed to misspellings during training. Therefore, even for large datasets using *BPE-dropout* can result in substantially better quality for practical applications where input is likely to be noisy.

7 Related work

Closest to our work in motivation is the work by Kudo (2018), who introduced the subword regularization framework multiple segmentation candidates and a new segmentation algorithm. Other segmentation algorithms include Creutz and Lagus (2006), Schuster and Nakajima (2012), Chitnis and DeNero (2015), Kunchukuttan and Bhattacharyya (2016), Wu and Zhao (2018), Banerjee and Bhattacharyya (2018).

Regularization techniques are widely used for training deep neural networks. Among regularizations applied to a network weights the most popular are Dropout (Srivastava et al., 2014) and L_2 regularization. Data augmentation techniques in natural language processing include dropping tokens at random positions or swapping tokens at close positions (Iyyer et al., 2015; Artetxe et al., 2018; Lample et al., 2018), replacing tokens at random positions with a placeholder token (Xie et al., 2017), replacing tokens at random positions with a token sampled from some distribution (e.g., based on token frequency or a language model) (Fadaee et al., 2017; Xie et al., 2017; Kobayashi, 2018). While *BPE-dropout* can be thought of as a regularization, our motivation is not to make a model robust by injecting noise. By exposing a model to different segmentations, we want to teach it to better understand the composition of words as well as subwords, and make it more flexible in the choice of segmentation during inference.

Several works study how translation quality depends on a level of granularity of a segmentation (Cherry et al., 2018; Kreutzer and Sokolov, 2018; Ding et al., 2019). Cherry et al. (2018) show that trained long enough character-level models tend to have better quality, but it comes with the increase of computational cost for both training and inference. Kreutzer and Sokolov (2018) find that, given flexibility in choosing segmentation level, the model prefers to operate on (almost) character level. Ding et al. (2019) explore the effect of BPE vocabulary size and find that it is better to use small vocabulary for low-resource setting and large vocabulary for a high-resource setting. Following these observations, in our experiments we use different vocabulary size depending on a dataset size to ensure the strongest baselines.

8 Conclusions

We introduce *BPE-dropout* – simple and effective subword regularization, which operates within the standard BPE framework. The only difference from BPE is how a word is segmented during model training: *BPE-dropout* randomly drops some merges from the BPE merge table, which results in different segmentations for the same word. Models trained with *BPE-dropout* (1) outperform BPE and the previous subword regularization on a wide range of translation tasks, (2) have better quality of learned embeddings, (3) are more robust to noisy input. Future research directions include adaptive dropout rates for different merges and an in-depth analysis of other pathologies in learned token embeddings for different segmentations.

Acknowledgments

We thank anonymous reviewers for the helpful feedback, Rico Sennrich for valuable comments on the first version of this paper, and Yandex Machine Translation team for discussions and inspiration.

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 955–962, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Unsupervised statistical machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3632–3642, Brussels, Belgium. Association for Computational Linguistics.

Tamali Banerjee and Pushpak Bhattacharyya. 2018. Meaningless yet meaningful: Morphology grounded subword-level NMT. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 55–60, New Orleans. Association for Computational Linguistics.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, Hong Kong. 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.

Rohan Chitnis and John DeNero. 2015. Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2088–2093, Lisbon, Portugal. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2006. Morfessor in the morpho challenge. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, pages 12–17. Citeseer.

Shuoyang Ding, Adithya Renduchintala, and Kevin Duh. 2019. A call for prudent choice of subword merge operations in neural machine translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 204–213, Dublin, Ireland. European Association for Machine Translation.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Frage: Frequency-agnostic word representation. In *Advances in neural information processing systems*, pages 1334–1345.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Julia Kreutzer and Artem Sokolov. 2018. Learning to segment inputs for nmt favors character-level processing. In *Proceedings of the 15th International Workshop on Spoken Language Translation*.

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.

Anoop Kunchukuttan and Pushpak Bhattacharyya. 2016. Orthographic syllable as basic unit for SMT between related languages. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1912–1917, Austin, Texas. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Myle Ott, Michael Auli, David Grangier, and MarcAurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning*, pages 3956–3965.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, Copenhagen, Denmark. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on*

Machine Translation: Research Papers, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

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

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.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

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

Yingting Wu and Hai Zhao. 2018. Finding better subword segmentation for neural machine translation. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 53–64. Springer.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lvy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *International Conference on Learning Representations*.

A Training time

Table 6 shows number of training batches for the experiments in Section 5.1 (Table 2), Table 7 — for the experiments in Section 5.2 (Table 3).

B Additional experiments

In the main text, all models were trained (and evaluated) on lowercased data. Here we provide results of the models trained and evaluated without lower case (Table 8).

	BPE	Kudo (2018)	*BPE-dropout*
IWSLT15			
En-Vi	23	26	36
Vi-En	23	29	33
En-Zh	30	29	43
Zh-En	39	51	100
IWSLT17			
En-Fr	36	45	60
Fr-En	32	46	85
En-Ar	30	60	62
Ar-En	41	51	59
WMT14			
En-De	468	450	501
De-En	447	442	525
ASPEC			
En-Ja	280	165	462
Ja-En	239	144	576

Table 6: Number of thousands of training batches for the experiments from Table 2.

	BPE	*BPE-dropout*		
		src-only	dst-only	both
250k	47	53	53	85
500k	160	210	250	320
1m	30	114	67	180
4m	100	321	180	600
16m	345	345	-	400

Table 7: Number of thousands of training batches for the experiments from Table 3. Note that we use batch size 4k tokens for small corpora (250k and 500k) and 32k tokens for large corpora (1m, 4m and 16m).

	BPE	*BPE-dropout*
IWSLT15		
En-Vi	31.44	**32.70**
Vi-En	32.19	**33.22**
IWSLT17		
En-Fr	38.79	**39.83**
Fr-En	38.06	**38.60**
En-Ar	14.30	**15.20**
Ar-En	31.56	**33.00**

Table 8: BLEU scores. Bold indicates the best score; differences with the baselines are statistically significant (with p-value of 0.05). (Statistical significance is computed via bootstrapping (Koehn, 2004).)