Approaches for Subword Regularization

Yongchang Hao

Aug 6, 2020

Neural Machine Translation of Rare Words with Subword Units

A quick review

- The standard subword model: BPE
- At each merge steps:
  1. count current pairs in dict
  2. merge the most frequent pair in dict
- BPE is deterministic

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates - Taku Kudo - ACL 2018

Intuition

- Use the segmentation ambiguity as noise to improve robustness
-

Subwords (_ means spaces)	Vocabulary id sequence
_Hell/o/_world	13586 137 255
_H/ello/_world	320 7363 255
_He/llo/_world	579 10115 255
_/He/l/l/o/_world	7 18085 356 356 137 255
_H/el/l/o/_/world	320 585 356 137 7 12295

Table 1: Multiple subword sequences encoding the same sentence "Hello World"

Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates

Approach - Training

- Induce the segmentation distribution P_d to the original loss

- $$\mathcal{L}_m(\theta) = \sum_{s=1}^{|D|} \mathbb{E}_{\substack{x \sim P_d(x|X^{(s)}) \\ y \sim P_d(y|Y^{(s)})}} \left[\log P(y|x;\theta)\right]$$

- Connect the loss function with subword segmentation

Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates

Approach - Inference

- one-best decoding:
  - Only translate $\underset{x}{\operatorname{argmax}} P_d(x|X)$
- n-best decoding:
  - Translate n best $P(x|X)$, select one with maximal $\dfrac{\log P(y|x)}{|y|^\lambda}$

Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates

Main result

Corpus	Language pair	baseline (BPE)	Proposed (one-best decoding)			Proposed (n-best decoding, n=64)		
			$l = 1$	$l = 64$ $\alpha = 0.1$	$l = \infty$ α=0.2/0.5	$l = 1$	$l = 64$ $\alpha = 0.1$	$l = \infty$ α=0.2/0.5
IWSLT15	en → vi	25.61	25.49	27.68*	27.71*	25.33	28.18*	28.48*
	vi → en	22.48	22.32	24.73*	26.15*	22.04	24.66*	26.31*
	en → zh	16.70	16.90	19.36*	20.33*	16.73	20.14*	21.30*
	zh → en	15.76	15.88	17.79*	16.95*	16.23	17.75*	17.29*
IWSLT17	en → fr	35.53	35.39	36.70*	36.36*	35.16	37.60*	37.01*
	fr → en	33.81	33.74	35.57*	35.54*	33.69	36.07*	36.06*
	en → ar	13.01	13.04	14.92*	15.55*	12.29	14.90*	15.36*
	ar → en	25.98	27.09*	28.47*	29.22*	27.08*	29.05*	29.29*
KFTT	en → ja	27.85	28.92*	30.37*	30.01*	28.55*	31.46*	31.43*
	ja → en	21.37	21.46	22.33*	22.04*	21.37	22.47*	22.64*
ASPEC	en → ja	40.62	40.66	41.24*	41.23*	40.86	41.55*	41.87*
	ja → en	26.51	26.76	27.08*	27.14*	27.49*	27.75*	27.89*
WMT14	en → de	24.53	24.50	25.04*	24.74	22.73	25.00*	24.57
	de → en	28.01	28.65*	28.83*	29.39*	28.24	29.13*	29.97*
	en → cs	25.25	25.54	25.41	25.26	24.88	25.49	25.38
	cs → en	28.78	28.84	29.64*	29.41*	25.77	29.23*	29.15*

Adversarial Subword Regularization for Robust Neural Machine Translation - Jungsoo Park, Mujeen Sung, Jinhyuk Lee, Jaewoo Kang - arXiv 2020

Intuition

- Generate subword segmentation that maximize the loss
- $\hat{x} = \underset{x \in \Omega(X_i)}{\operatorname{argmax}} \left((\nabla_{\tilde{x}} \mathcal{L})^T \cdot (emb(x) - emb(\tilde{x}) \right)$
  - Ω denotes all possible segmentation, x denotes a real word where consists of subwords $x = [x_1, \ldots x_k]$, $emb(x) = \frac{1}{k} \sum_{i=1}^{k} emb(x_i)$
  - $x \in \Omega(X_i)$ is done by Kudo's work.

Adversarial Subword Regularization for Robust Neural Machine Translation

Main result

Lang Pair	**BASE**	**SR**	**ADVSR**
	IWSLT17		
FR $\rightarrow$ EN	37.9	38.1	**38.7**
EN $\rightarrow$ FR	38.8	39.1	**39.9**
AR $\rightarrow$ EN	31.7	32.3	**33.3**
EN $\rightarrow$ AR	**14.4**	**14.3**	**14.7**
	IWSLT15		
CS $\rightarrow$ EN	28.9	30.5	**32.0**
EN $\rightarrow$ CS	20.4	21.7	**23.9**
VI $\rightarrow$ EN	28.1	**28.4**	**29.0**
EN $\rightarrow$ VI	30.9	31.7	**32.3**
	IWSLT13		
PL $\rightarrow$ EN	19.1	19.7	**20.9**
EN $\rightarrow$ PL	13.5	14.1	**15.1**
TR $\rightarrow$ EN	21.3	22.6	**23.4**
EN $\rightarrow$ TR	12.6	**14.4**	**14.0**

Adversarial Subword Regularization for Robust Neural Machine Translation

Robustness analysis

Dataset	BASE	SR	ADVSR
MTNT2018			
FR → EN	25.7	**27.6**	**27.2**
EN → FR	26.7	27.5	**28.2**
MTNT2018 + FT			
FR → EN	36.5	37.9	**38.8**
EN → FR	33.2	34.4	**35.3**
MTNT2019			
FR → EN	27.6	29.3	**30.2**
EN → FR	22.8	**23.8**	**24.1**
MTNT2019 + FT			
FR → EN	36.2	**38.1**	**38.6**
EN → FR	27.6	28.2	**28.9**

Table 3: BLEU scores on the out-of-domain MTNT dataset. **FT** denotes finetuning with the MTNT2018 training dataset.

Method	0.1	0.2	0.3	0.4	0.5
FR → EN					
BASE	30.7	25.6	20.3	16.2	11.4
SR	33.2	28.5	23.3	18.7	14.7
ADVSR	**34.8**	**32.0**	**29.2**	**25.7**	**22.2**
EN → FR					
BASE	31.1	24.2	18.6	14.6	10.6
SR	34.2	27.8	23.9	18.9	14.4
ADVSR	**35.1**	**30.3**	**26.4**	**23.0**	**19.1**

Table 4: BLEU scores on the synthetic dataset of typos. The column lists results for different noise fractions.

Drawbacks of these 2 papers

Too complicated

- EM algorithm to optimize vocabulary
- Viterbi algorithm to find best segmentation
- Enhanced Suffix Array to find frequent substring
- Forward-DP Backward-A* in l-best search
- Forward-Filtering Backward-Sampling for infinity search
- ......

BPE-Dropout: Simple and Effective Subword Regularization - Ivan Provilkov, Dmitrii Emelianenko, Elena Voita - ACL 2020

Approach

- At each merge steps:
  1. count current pairs in dict
  2. randomly drop some merge operations with the probability p
  3. merge the most frequent pair in dict
- BPE-dropout is stochastic

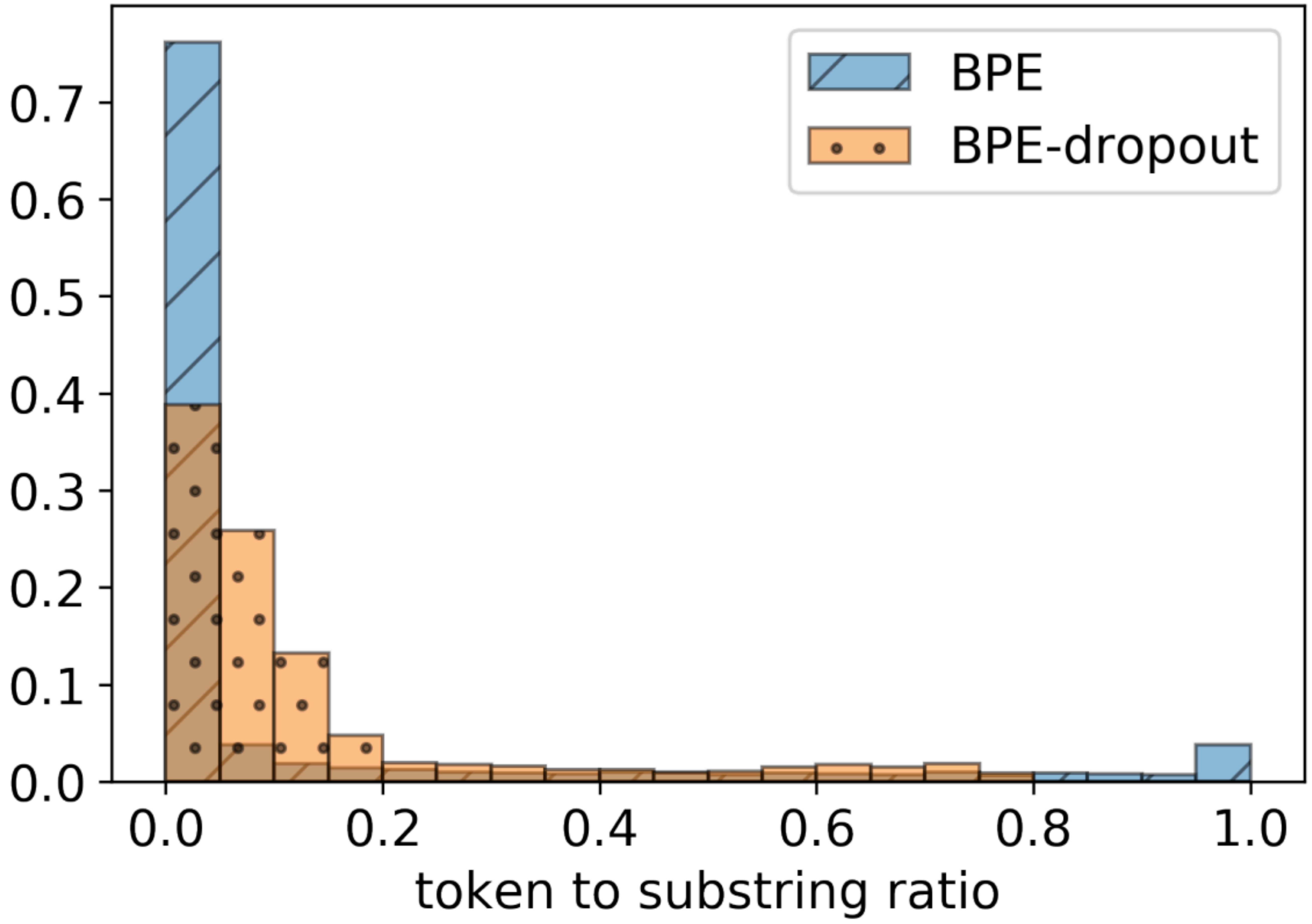
BPE-Dropout: Simple and Effective Subword Regularization

Main result

	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**

BPE-Dropout: Simple and Effective Subword Regularization

#token / #substring is smoother



BPE-Dropout: Simple and Effective Subword Regularization

Embedding is more reasonable

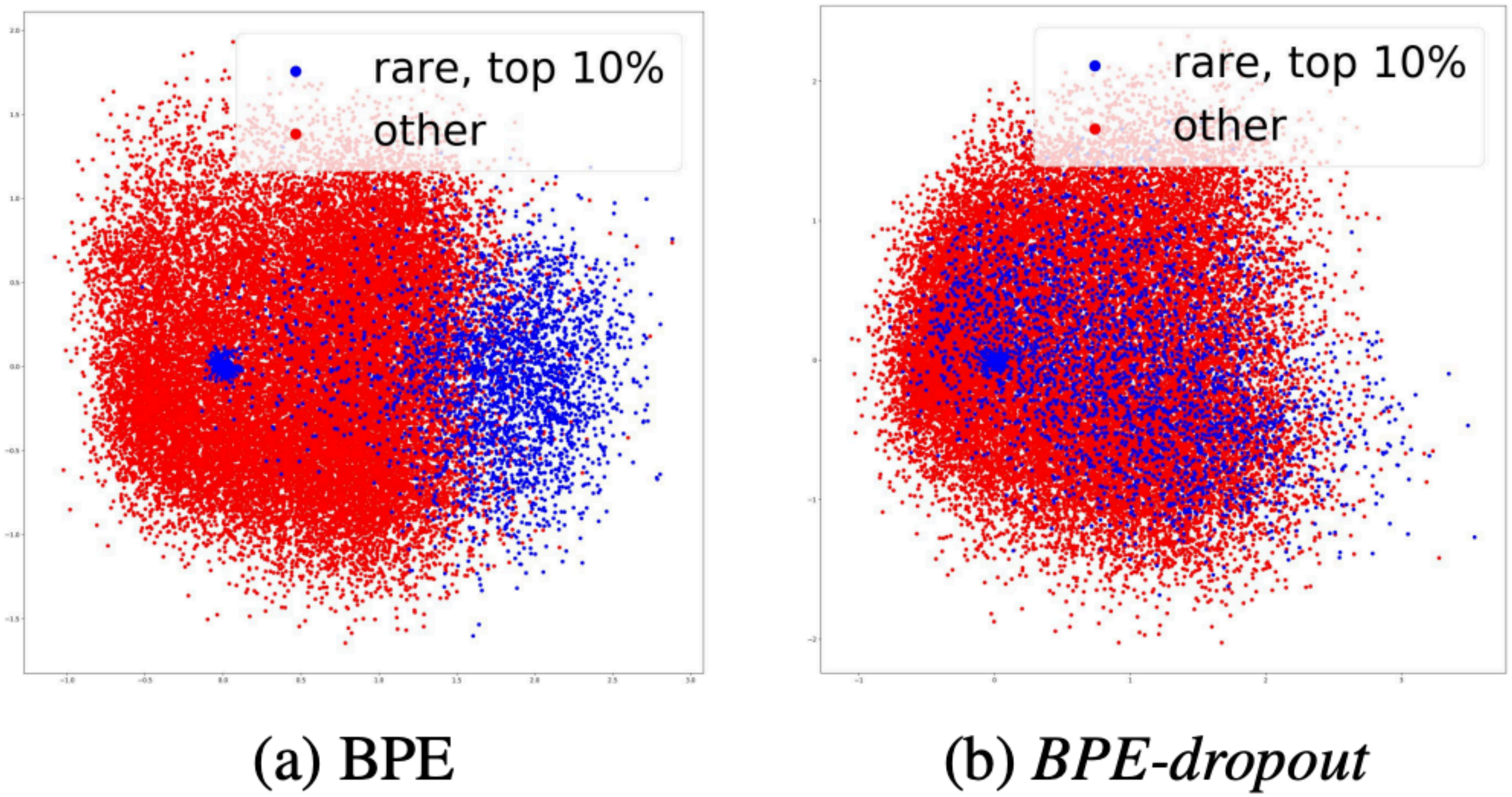


Figure 7: Visualization of source embeddings. Models trained on WMT14 En-Fr (4m).

BPE-Dropout: Simple and Effective Subword Regularization

More robust to misspelled input

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

Conclusion

	Easy to Implement	Contribution
Subword Regularization	No	Induce probability to subword segmentation
Adversarial Subword Regularization	No	Integrate adversarial into subword segmentation
BPE-dropout	Yes	Introduce randomness in BPE merge operations

Thank You