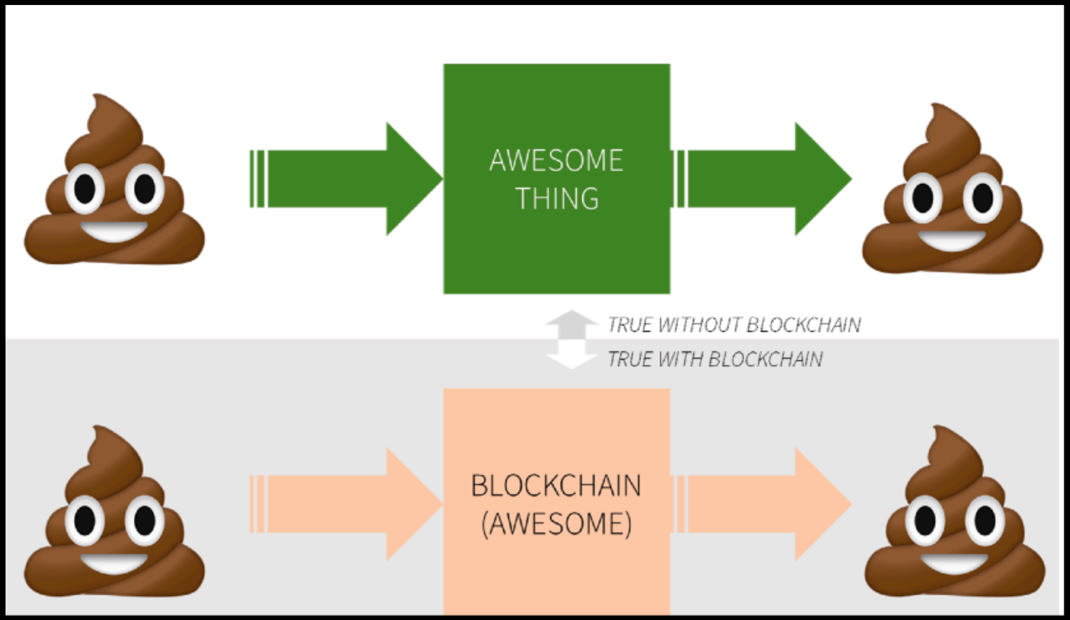
BPE-Dropout
Simple and Effective Subword Regularization

I Provilkov

Kim Soo Jung

왜 이 논문을 선택?



자연어 처리에 관심이 많다(NLP lover)
모델만 중요한게 아니더라
garbage in garbage out은 불변의 법칙
데이터부터 조진다.

근데 도메인 특성상 새로운 단어를 많이 본다
새로운 단어는 이미 만들어 놓은 사전에 없어서
out of vocabulary 라는 문제가 생기게 되고
이를 해결하기 위해 rare word, word segmentation에 관심을 갖게 되었다.

용어정리

단어들의 집합 (vocabulary), 사전에 정의, 기계가 외움

이러한 Vocab에 없는 단어가 등장하면? OOV(out-of-vocabulary)

즉, 단어 집합에 없는 단어는 UNK(unkown word), rare word

용어정리

풀 수 는 있는거야...?

그럼 이런 OOV문제를 어떻게 풀지?

용어정리

우리의 소중한 모델이 아직 배운적이 없는 단어라도 대처할 수 있게
Subword Segmentation, 내부 단어를 분리

그 기법으로 BPE(Byte pair encoding)과
WPM(word piece model)이 있는데

오늘은 BPE관련!

BPE(Byte pair encoding)

- Neural Machine Translation of Rare Words with Subword Unit, R Sennrich, 2016
기존에 이쓴 단어를 분리하는 알고리즘
글자(char) 단위에서 점차적으로 단어 집합(vocab)을 만들어 내는 방식

train set에 있는 단어와 단어의 빈도수
low: 5
lower : 2
newest : 6
widest : 3

Vocab , 데이터 셋에서 중복 제거
low
lower
newest
widest

만약 lowest가 나온다면 ???

BPE(Byte pair encoding) 적용

```python
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)

print(vocab)
```

\# Dict
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

\# Vocab
l, o, w, e, r, n, w, s, t, i, d

vocab

1. 맨 뒤에 특수기호 '</w>' 를 넣음.
2. 한 글자(char) 단위로 모두 띄어 초기화.
3. vocab의 value는 빈도수.
   - low는 5번
   - newest는 6번

BPE(Byte pair encoding) 적용

```python
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)

print(vocab)
```

best = max(pairs, key=pairs.get)

- 빈도수가 가장 많은 bi-gram을 찾음.
- 찾은 bi-gram을 하나의 unit으로 merge.
- num_merge만큼 반복

BPE(Byte pair encoding) 적용

```
iterating 1 / 10 ...defaultdict(<class 'int'>,
            {('d', 'e'): 3,
             ('e', 'r'): 2,
             ('e', 's'): 9,
             ('e', 'w'): 6,
             ('i', 'd'): 3,
             ('l', 'o'): 7,
             ('n', 'e'): 6,
             ('o', 'w'): 7,
             ('r', '_'): 2,
             ('s', 't'): 9,
             ('t', '_'): 9,
             ('w', '_'): 5,
             ('w', 'e'): 8,
             ('w', 'i'): 3})

best: ('e', 's')
```

update Dict
l o w </w>: 5
l o w e r </w> : 2
n e w es t </w> : 6
w i d es t </w> : 3

iter1
update Vocab
l, o, w, e, r, n, w, s, t, i, d, es

```
iterating 2 / 10 ...defaultdict(<class 'int'>,
            {('d', 'es'): 3,
             ('e', 'r'): 2,
             ('e', 'w'): 6,
             ('es', 't'): 9,
             ('i', 'd'): 3,
             ('l', 'o'): 7,
             ('n', 'e'): 6,
             ('o', 'w'): 7,
             ('r', '_'): 2,
             ('t', '_'): 9,
             ('w', '_'): 5,
             ('w', 'e'): 2,
             ('w', 'es'): 6,
             ('w', 'i'): 3})

best: ('es', 't')
```

update Dict
l o w </w>: 5
l o w e r </w> : 2
n e w est </w> : 6
w i d est </w> : 3

iter2
Vocab
l, o, w, e, r, n, w, s, t, i, d, es, est

BPE(Byte pair encoding) 적용

원하는 단어 집합의 크기가 될 때까지 반복!
iteration 설정, 논문에서는 10번

```
# update Dict
low </w>:  5
low e r </w> : 2
newest </w> : 6
widest </w> : 3
```

```
# iter10
# update Vocab
l, o, w, e, r, n, w, s, t, i, d, es, est, lo, low, ne, new, newest, wi, wid, widest
```

다시, lowest라는 단어가 등장한다면??

1. lowest를 char 단위로 분할 -> l, o, w, e, s, t
2. low와 est를 찾아냄
3. lowest -> low와 est로 인코딩

결론적으로

BPE는
빈번히 등장하는 substring을 단어로 학습하고,
자주 등장하지 않는 단어들을 최대한 의미보존을 할 수 있는 최소한의 units로 표현

즉, 자주 이용되는 단어는 그 자체가 unit이 되며, rare words가 subword unit으로 나누어짐

BPE-Dropout: Simple and Effective Subword Regularization

기존의 BPE 말이야.. 다 좋아 다 좋은데

common word는 유지하고, rare word/ UNK 는 subword로 표현할 때
각 단어가 오직 하나의 segmentation만 나오던데...

이렇게 하면 not to reach full potential !!
그래서 우리는 bpe 이용해서 multiple segmentation하려고해

근데 multiple sementation candidates하는 논문이 있긴 있다???
근데 이거 bpe 못쓰고, 구현 어렵고 복잡함ㅋ

BPE-dropout

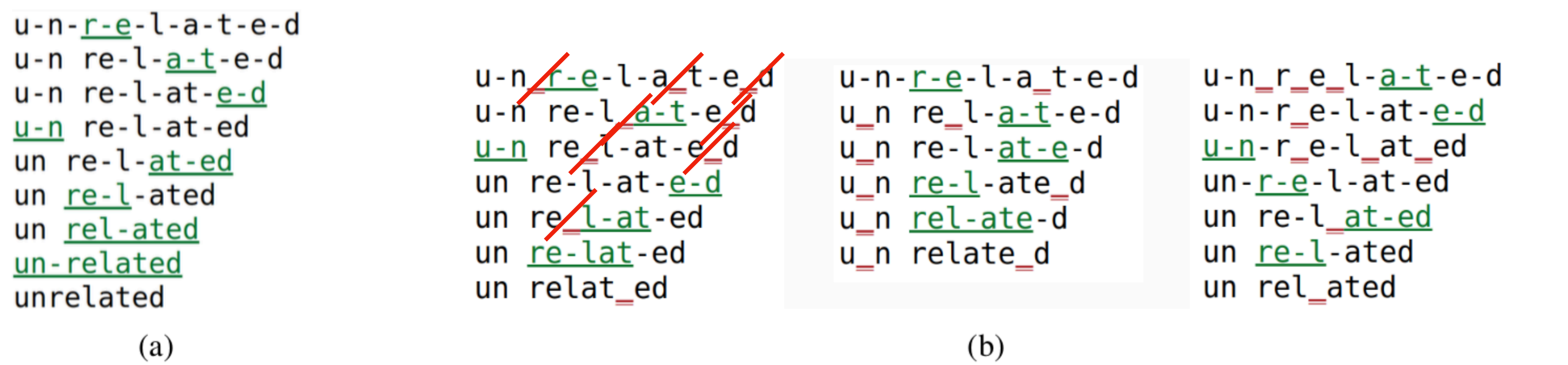


Figure 1: Segmentation process of the word '*unrelated*' using (a) BPE, (b) *BPE-dropout*. Hyphens indicate possible merges (merges which are present in the merge table); merges performed at each iteration are shown in green, dropped – in red.

각 merge하는 step에서 random하게 drop해줌

when word = unrelated,

if BPE : unrelated (오오직 1개)
if BPE-Dropout : un relate_ed , u_n relate_d, un rel_ated (multiple segmentation)

BPE-dropout algorithm

Algorithm 1: BPE-dropout

```
current_split ← characters from input_word;
do
    merges ← all possible merges 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;
```

training 할 때
probability p(p=0.1)를 적용하여
model이 different segmentations에 노출될 수 있도록

Experiments settings

Model

NMT system is Transformer base
- Attention is all you need

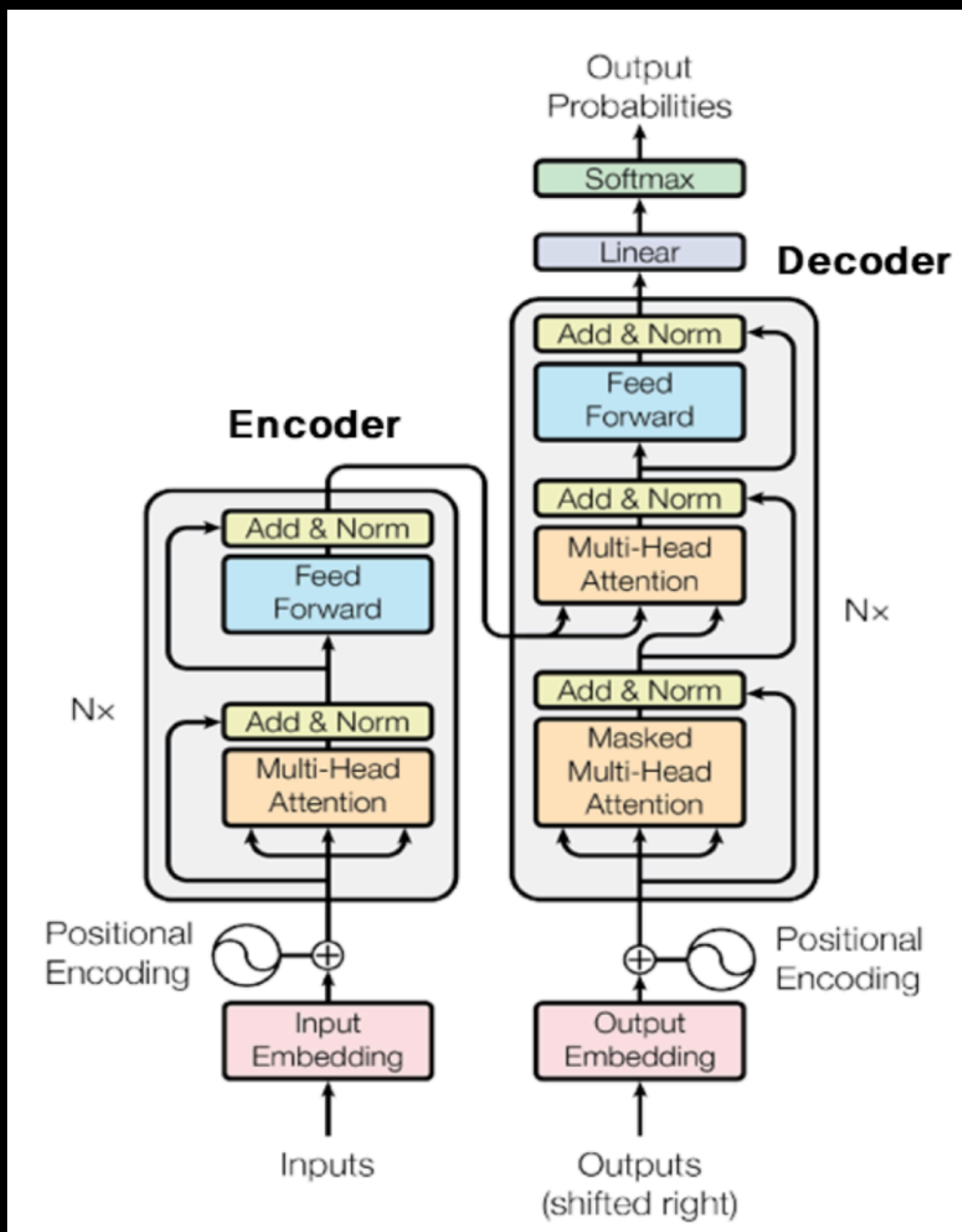


Figure 1: The Transformer - model architecture.

machine translation dataset

		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. (We explain the choice of the value of p for *BPE-dropout* in Section 5.3.)

IWSLT : TED and TEDx talks
WMT : news commentaries and parliament proceedings

Inference

using 1-best decoding

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.

NMT Results

	BPE	Kudo (2018)	*BPE-dropout*
IWSLT15			
En-Vi	31.78	32.43	**33.27**
Vi-En	30.83	32.36	**32.99**
En-Zh	21.07	**23.15**	**23.27**
Zh-En	18.29	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	43.69	**44.92**	44.19
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).)

뭐, 다 좋다

근데 Chinese랑 Japanese는 잘 안나옴
no explicit word boundaries

그리고 Kudo는 다른 세그멘테이션 기법씀
우리는 BPE에서만 쓴거구!!
그래서 충분히 더 끌올할 수 있어!

여러 실험 결과들

src-only good!

	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**

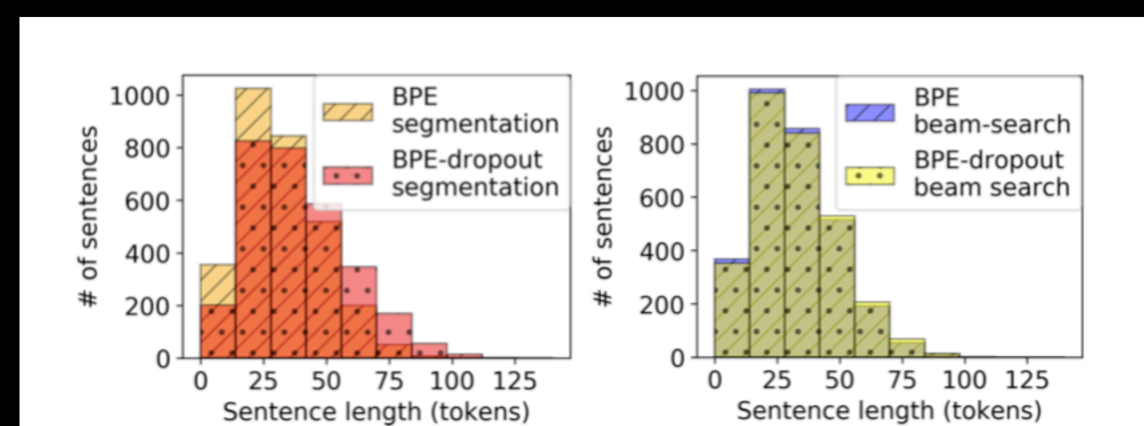
Table 3: BLEU scores for models trained with *BPE-dropout* on a single side of a translation pair or on both sides. Models trained on random subsets of WMT14 En-Fr dataset. Bold indicates the best score and all scores whose difference from the best is not statistically significant (with p-value of 0.05).

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.

Inference time

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

1. not to tune vocab size for each dataset
2. choose vocab size depending on desired
   - small vocab
     beneficial # of params
   - large vocab
     beneficial inference time



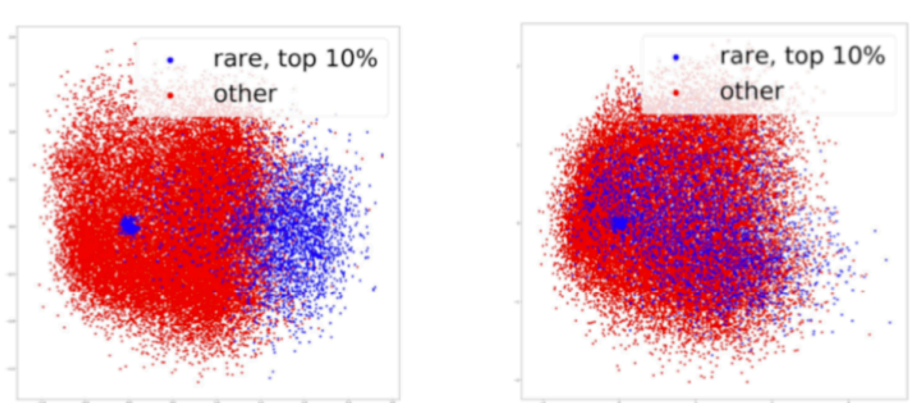
Results

original token

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

Applied BPE-dropout, NN token are share sequences of char with original token



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

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

Properties of the learned embeddings

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

Results

Robustness 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			
original	33.38	**33.85**	+0.47
misspelled	30.30	**32.13**	+1.83

Table 5: BLEU scores for models trained on WMT14 dataset evaluated given the original and misspelled source. For En-Fr, models were trained on 4m randomly chosen sentence pairs.

모델은 훈련중에 misspelled에 노출되지 않음

* misspelled
  - removal of one char from word
  - insertion of a random char into word
  - substitution

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.

Conclusion

Different from BPE：randomly drops some merges from BPE merge table

1. Outperform BPE and subword regularization on translation task
2. Have better quality of learned embeddings
3. More robust to noisy input


Thanks :)