SKKU

Neural Machine Translation of Rare Words with Subword Unit

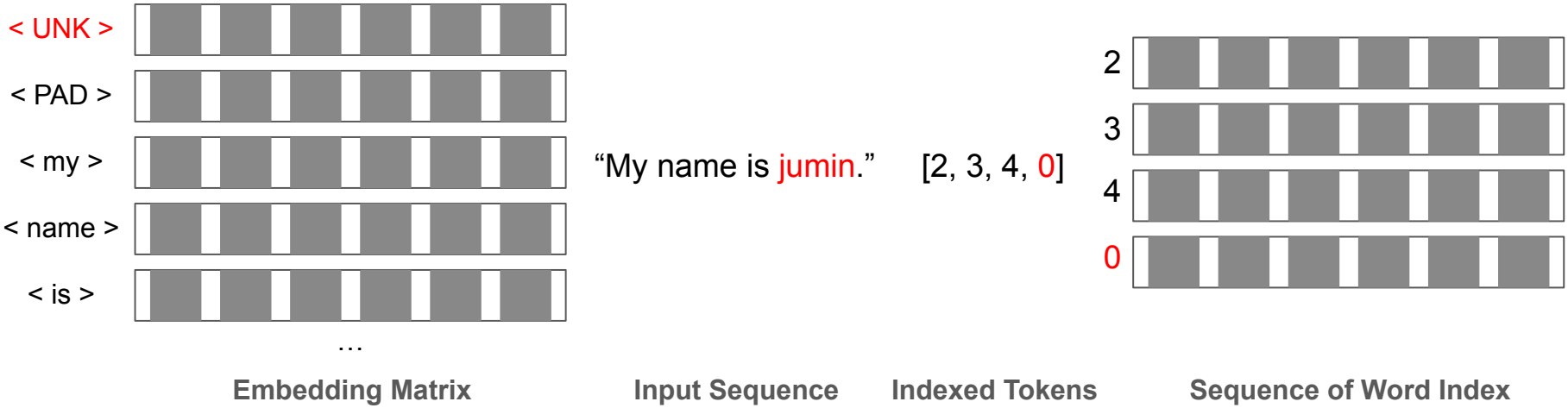
Natural Language Processing

skku 김정훈

skku 이주민

Motivation

- Translation Model은 Open-Vocabulary Problem을 가짐
- Tokenizing은 NLP tasks에서 성능에 중요한 영향을 끼침
- 이 논문에선 Out Of Vocabulary(OOV) problem 해결하고자 함
    - Input sequence에 Dictionary에 없는 단어를 OOV(Out-Of-Vocabulary) 또는 UNK(Unknown Token)이라 표현하는데, 이때 이런 단어로 인해 문제가 어려워지는 상황



3. Subword Translation

Subword Translation

- 이름 등의 **고유 명사**
  - 음절 별로 대응
- 동의어, 외래어 등 같은 **Origin을 갖는 단어**
  - 일정한 규칙을 갖고 변형되므로, Character-level Translation 사용
- **복합어**
  - Subword를 번역한 후 결합

위와 같은 규칙으로 German Training Data에서 가장 빈도 낮은 100개의 Word를 분석하면 English Data를 통해 56개의 복합어, 21개의 고유명사, 6개의 외래어 등을 찾아냄

즉, 새로운 단어여도 Subword Unit의 조합으로 구성된 경우가 많으므로,
한 단어를 여러 Subword Unit으로 분리해서 Embedding하는 전처리 작업으로 번역 기능 향상 가능

3.1 Related Work

OOV Problem 해결을 위한 시도

- **Back-Off Dictionary**
  - **Copying Mechanism** = Dictionary look-up

: Source Sentence와 Target Sentence가 있을 때, Target Sentence에 있는 OOV단어를 Source Sentence에서 찾아 그대로 복사하는 것

  - **Transliteration**

: Target Sentence에 있는 OOV단어를 발음되는 대로 번역하는 것

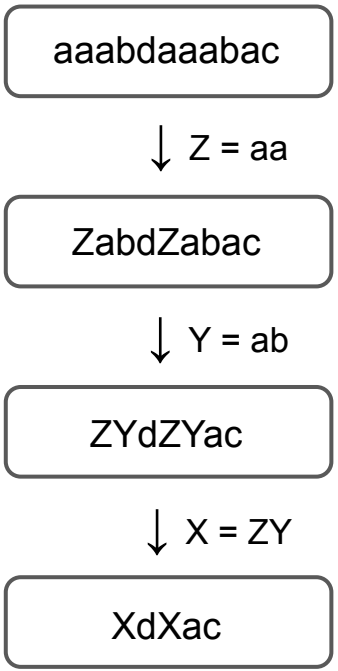
- **Subword Units**

- Character
- mixed Character and word
- **Byte Pair Encoding(BPE)**

3.2 Byte Pair Encoding (BPE)

- **BPE(Byte pair encoding)** 알고리즘

  : 빈도수가 높은 byte pair를 사용되지 않은 byte로 교체하는 데이터 압축 알고리즘



Byte Pair Encoding

- **pair of bytes → pair of characters**
- Bottom up 방식

  : Character 단위에서 점차 Vocabulary 생성

- **Training**

1. Training Dataset에 있는 단어들을 모두 Character 또는 Unicode 단위로 Vocabulary 생성
2. 빈도수가 가장 높은 unigram 통합
3. 이 과정을 정해진 Vocabulary 크기가 될 때까지 반복

- **Test**

  처음보는 단어를 Character 단위로 분리 후,

  Vocabulary 내 가장 큰 Subword 단위로 합침

“ Unsupervised, Multi-lingual Text Tokenizer ”

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)
```

Iteration 1

Frequency of pairs : {('l', 'o'): 7, ('o', 'w'): 7, ('w', '</w>'): 5, ('w', 'e'): 8, ('e', 'r'): 2, ('r', '</w>'): 2, ('n', 'e'): 6, ('e', 'w'): 6, ('e', 's'): 9, ('s', 't'): 9, ('t', '</w>'): 9, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'e'): 3}
New merge: ('e', 's')
Dictionary: {'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}

Iteration 2

Frequency of pairs : {('l', 'o'): 7, ('o', 'w'): 7, ('w', '</w>'): 5, ('w', 'e'): 2, ('e', 'r'): 2, ('r', '</w>'): 2, ('n', 'e'): 6, ('e', 'w'): 6, ('w', 'es'): 6, ('es', 't'): 9, ('t', '</w>'): 9, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'es'): 3}
New merge: ('es', 't')
Dictionary: {'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w est </w>': 6, 'w i d est </w>': 3}

Iteration 3

Frequency of pairs : {('l', 'o'): 7, ('o', 'w'): 7, ('w', '</w>'): 5, ('w', 'e'): 2, ('e', 'r'): 2, ('r', '</w>'): 2, ('n', 'e'): 6, ('e', 'w'): 6, ('w', 'est'): 6, ('est', '</w>'): 9, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est'): 3}
New merge: ('est', '</w>')
Dictionary: {'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w est</w>': 6, 'w i d est</w>': 3}

Iteration 4

Frequency of pairs : {('l', 'o'): 7, ('o', 'w'): 7, ('w', '</w>'): 5, ('w', 'e'): 2, ('e', 'r'): 2, ('r', '</w>'): 2, ('n', 'e'): 6, ('e', 'w'): 6, ('w', 'est</w>'): 6, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est</w>'): 3}
New merge: ('l', 'o')
Dictionary: {'lo w </w>': 5, 'lo w e r </w>': 2, 'n e w est</w>': 6, 'w i d est</w>': 3}

Iteration 5

Frequency of pairs : {('lo', 'w'): 7, ('w', '</w>'): 5, ('w', 'e'): 2, ('e', 'r'): 2, ('r', '</w>'): 2, ('n', 'e'): 6, ('e', 'w'): 6, ('w', 'est</w>'): 6, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est</w>'): 3}
New merge: ('lo', 'w')
Dictionary: {'low </w>': 5, 'low e r </w>': 2, 'n e w est</w>': 6, 'w i d est</w>': 3}

Iteration 6

Frequency of pairs : {('low', '</w>'): 5, ('low', 'e'): 2, ('e', 'r'): 2, ('r', '</w>'): 2, ('n', 'e'): 6, ('e', 'w'): 6, ('w', 'est</w>'): 6, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est</w>'): 3}
New merge: ('n', 'e')
Dictionary: {'low </w>': 5, 'low e r </w>': 2, 'ne w est</w>': 6, 'w i d est</w>': 3}

Iteration 7

Frequency of pairs : {('low', '</w>'): 5, ('low', 'e'): 2, ('e', 'r'): 2, ('r', '</w>'): 2, ('ne', 'w'): 6, ('w', 'est</w>'): 6, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est</w>'): 3}
New merge: ('ne', 'w')
Dictionary: {'low </w>': 5, 'low e r </w>': 2, 'new est</w>': 6, 'w i d est</w>': 3}

Iteration 8

Frequency of pairs : {('low', '</w>'): 5, ('low', 'e'): 2, ('e', 'r'): 2, ('r', '</w>'): 2, ('new', 'est</w>'): 6, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est</w>'): 3}
New merge: ('new', 'est</w>')
Dictionary: {'low </w>': 5, 'low e r </w>': 2, 'newest</w>': 6, 'w i d est</w>': 3}

Iteration 9

Frequency of pairs : {('low', '</w>'): 5, ('low', 'e'): 2, ('e', 'r'): 2, ('r', '</w>'): 2, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est</w>'): 3}
New merge: ('low', '</w>')
Dictionary: {'low</w>': 5, 'low e r </w>': 2, 'newest</w>': 6, 'w i d est</w>': 3}

Iteration 10

Frequency of pairs : {('low', 'e'): 2, ('e', 'r'): 2, ('r', '</w>'): 2, ('w', 'i'): 3, ('i', 'd'): 3, ('d', 'est</w>'): 3}
New merge: ('w', 'i')
Dictionary: {'low</w>': 5, 'low e r </w>': 2, 'newest</w>': 6, 'wi d est</w>': 3}

New Word : lowest ➡️ low , est

4. Evaluation

BPE를 적용한 2가지 방법

1. **Source / Target 각각** Encoding 학습

- 장점 : 각각의 Subword Unit이 각 언어의 Training Text에 반드시 존재
- 단점 : 각각의 언어에서 동일한 단어가 다른 방식으로 Segmented 될 수 있어 Neural Model이 Subword Units를 Mapping하기 어려움

2. **Joint BPE**

: **Source / Target을 합쳐서** 하나의 Encoding 학습

- 장점 : Source와 Target Segmentation 사이에 일관성 존재
- 단점 : 1 방법에 비해 Text와 Vocabulary 사이즈가 Compact 하지 못함

4.1 Subword statistics

Subword Statistics

tokens : Text size
types : Vocabulary size
UNK : OOV word 개수

segmentation	# tokens	# types	# UNK
none	100 m	1 750 000	1079
characters	550 m	3000	0
character bigrams	306 m	20 000	34
character trigrams	214 m	120 000	59
compound splitting△	102 m	1 100 000	643
morfessor*	109 m	544 000	237
hyphenation◇	186 m	404 000	230
BPE	112 m	63 000	0
BPE (joint)	111 m	82 000	32
character bigrams (shortlist: 50 000)	129 m	69 000	34

Table 1: Corpus statistics for German training corpus with different word segmentation techniques. #UNK: number of unknown tokens in newstest2013. △: (Koehn and Knight, 2003); *: (Creutz and Lagus, 2002); ◇: (Liang, 1983).

- BPE를 사용한 경우, #UNK가 0개임
- BPE Joint의 경우에도 32개로 좋은 성능을 보임

4.2 Translation experiments

- 2가지 **Dataset** 사용 : English → German, English → Russian

- **성능 지표**

- BLEU : 기계 번역 결과와 사람이 직접 번역한 결과가 얼마나 유사한지 n-gram을 기반으로 비교하여 번역 성능 측정
- CHRF3(Character n-gram F3 Score) : F1 score(combining tri-gram precision and recall)
- Unigram F1 Score : BLEU unigram(brevity penalty 제외)와 Recall의 조합

- **Baseline**

- WUnk : back-off dictionary를 사용하지 않고 처음보는 단어를 모두 <UNK>로 표기하는 model
- WDict : back-off dictionary를 사용한 model
- C2-50k : Character n-gram에 대한 baseline으로 Char-bigram을 사용한 모델
  Unigram의 경우 제대로된 open-vocabulary를 제공할 수 있지만, 성능이 좋지 않으므로 bigram을 baseline으로 지정

Translation Experiments

표 [English → German]

name	segmentation	shortlist	vocabulary source	vocabulary target	BLEU single	BLEU ens-8	CHRF3 single	CHRF3 ens-8	unigram F_1 (%) all	unigram F_1 (%) rare	unigram F_1 (%) OOV
syntax-based (Sennrich and Haddow, 2015)					24.4	-	55.3	-	59.1	46.0	37.7
WUnk	-	-	300 000	500 000	20.6	22.8	47.2	48.9	56.7	20.4	0.0
WDict	-	-	300 000	500 000	22.0	24.2	50.5	52.4	58.1	36.8	**36.8**
C2-50k	char-bigram	50 000	60 000	60 000	**22.8**	**25.3**	51.9	53.5	58.4	40.5	30.9
BPE-60k	BPE	-	60 000	60 000	21.5	24.5	**52.0**	53.9	58.4	40.9	29.3
BPE-J90k	BPE (joint)	-	90 000	90 000	**22.8**	24.7	51.7	**54.1**	**58.5**	**41.8**	33.6

표 [English → Russian]

name	segmentation	shortlist	vocabulary source	vocabulary target	BLEU single	BLEU ens-8	CHRF3 single	CHRF3 ens-8	unigram F_1 (%) all	unigram F_1 (%) rare	unigram F_1 (%) OOV
phrase-based (Haddow et al., 2015)					24.3	-	53.8	-	56.0	31.3	16.5
WUnk	-	-	300 000	500 000	18.8	22.4	46.5	49.9	54.2	25.2	0.0
WDict	-	-	300 000	500 000	19.1	22.8	47.5	51.0	54.8	26.5	6.6
C2-50k	char-bigram	50 000	60 000	60 000	**20.9**	**24.1**	49.0	51.6	55.2	27.8	17.4
BPE-60k	BPE	-	60 000	60 000	20.5	23.6	**49.8**	52.7	55.3	29.7	15.6
BPE-J90k	BPE (joint)	-	90 000	100 000	20.4	**24.1**	49.7	**53.0**	**55.8**	**29.7**	**18.3**

- BPE와 joint-BPE 모두 Baseline의 F1 score를 뛰어넘음
- BLEU나 CHRF3 score는 Rare word의 중요한 성능을 평가에 정확히 반영하지 못함에도 불구하고 BPE와 joint-BPE의 성능 향상이 있음
  - Rare word의 경우 문장의 중요한 의미를 담고 있는 경우가 많음

5. Analysis

5.1 Unigram accuracy

Unigram Accuracy

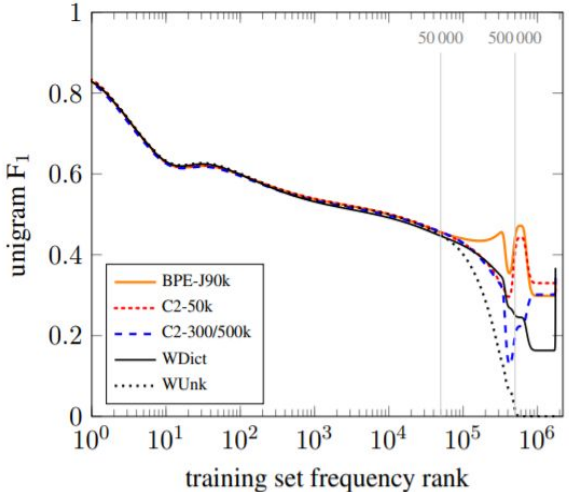


Figure 2: English→German unigram F_1 on newstest2015 plotted by training set frequency rank for different NMT systems.

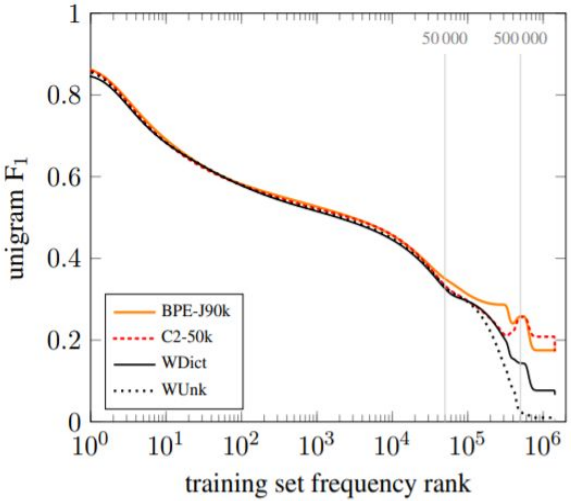


Figure 3: English→Russian unigram F_1 on newstest2015 plotted by training set frequency rank for different NMT systems.

- C2-3/500k : Vocabulary 집합 크기가 성능에 어떤 영향을 주는지 확인하고자 구축
- **Rare Word에 대한 unigram F1 점수** : Less Frequency 일수록 성능을 떨어짐
  - 성능 : BPE > C2-3/500k > WDict > WUnk
  - **BPE** : Less Sparse → vocabulary Size 감소 → 많은 단어 표현 가능 → 성능 증가

6. Conclusion

“ Unsupervised, Multi-lingual Text Tokenizer ”

- **Open-Vocabulary Translation** by representing **Rare and Unseen Words** as a sequence of **Subword Units**.
- More effective than using a **back-off translation model**

- 많은 NLP model은 text tokenizer with BPE algorithm 채택

- Attention is All You Need - EIMo - BERT - GPT-2