Subowrd Tokenizer

Japanese and korean voice search.

Schuster, Mike, and Kaisuke Nakajima.
ICASSP 2012

김수형

목차

- What is Subword Tokenizer
- BPE(Byte Pair Encoder)
- WPM & Unigram LM
- Which one should we use?

- **Subword Tokenizer**
  - ✓ 하나의 단어를 더 작은 단위의 의미 있는 여러 서브 워드로 나누는 작업
    - ➢ 예시) lower -> l, o, w, e, r, ow, er

      독도는 우리땅 -> 독, 도, 는, 우, 리, 땅, 독도, 우리
  - ✓ 언어의 특성에 구애 받지 않고 대부분의 언어를 Tokenize 가능
  - ✓ BPE, Word piece model, Unigram language model

- **BPE(Byte Pair Encoding)**
  - ✓ 반복되는 문자열을 하나의 문자로 합치는 데이터 압축 알고리즘
    - ➢ Aaabbbabab -> ab=X -> AaXbbXX
  - ✓ 모든 단어를 유니그램 단위로 분리
  - ✓ 가장 빈도수가 높은 유니그램의 쌍을 하나의 유니그램으로 통합

Subword Tokenizer

BPE(Byte Pair Encoding)

토큰	빈도
hug	10
pug	5
pun	12
bun	4
hugs	5

바이그램 쌍	빈도
b, u	4
g, s	5
h, u	15
p, u	17
u, g	20
u, n	16

Subword Tokenizer

WPM &Unigram LM

- **WPM(Word Piece Model)**
  - ✓ BPE를 기반으로 만들어진 Tokenizer
  - ✓ 코퍼스의 Likelihood를 가장 높이는 pair를 병합한다.
  - ✓ BERT를 훈련할 때 사용됨
- **Unigram Language Model**
  - ✓ 만들어진 모든 pair중 Likelihood가 낮은 pair를 제거함
  - ✓ Sentence piece의 default로 사용됨

- **WPM(Word Piece Model)**

1. 원하는 크기의 vocab_size를 정한다.
2. train_data의 단어들을 character들의 순서로 쪼갠다.
   - ❖ 띄어쓰기는 '_' 로 변환하여 첫번째 character와 함께 하나의 토큰으로 쓰인다.
   - ❖ E.g. high five -> h, i, g, h, _f, i, v, e
3. Pair로 만들었을 때, likelihood가 가장 높은 pair를 합쳐서 model에 추가한다.
4. 3번의 방법을 vocab_size가 될 때까지 반복한다.

- **Unigram Language Model**

1. 원하는 크기의 vocab_size를 정한다.
2. train_data의 단어들을 character들의 순서로 쪼갠다
   - ❖ 띄어쓰기는 '_' 로 변환하여 첫번째 character와 함께 하나의 토큰으로 쓰인다.
3. 만들 수 있는 모든 조합의 subwords를 생성한다.
4. Model을 likelihood로 정렬하였을 때, likelihood가 작은 10~20%의 subword를 model에서 제거한다.
5. 4번의 방법을 vocab_size가 될 때까지 반복한다.

Subword Tokenizer

Calculate Likelihood

Inference

Suppose we have D generated by some unigram model θ

- and we want to learn that are the parameters of θ: estimate $P(w \mid \theta)$ for each $w \in D$
- this is a Parameter Estimation Problem
- Maximum Likelihood Estimation (MLE) is a possible solution: $\hat{\theta} = \arg\max_{\theta} P(D \mid \theta)$
- since unigrams is a multinomial probability distribution, MLE is:
- $P(w \mid \hat{\theta}) = \frac{c(w, D)}{|D|}$ where $c(w, D)$ is the count of word w in document D and $|D|$ is the total number of words in the document

How this formula is derived?

- consider log likelihood function: $\log P(D \mid \theta) = \sum_{w \in V} c(w, D) \log P(w \mid \theta)$
- we want to maximize it s.t. $P(w \mid \theta)$ is a Probability Distribution i.e. $\sum_{w \in V} P(w \mid \theta) = 1$
- use Lagrange Multipliers to convert this constrained optimization problem into an unconstrained one
- so let $L(\theta, \lambda) = \log P(D \mid \theta) + \lambda (1 - \sum P(w \mid \theta)) = \sum_{w \in V} c(w, D) \log P(w \mid \theta) + \lambda (1 - \sum P(w \mid \theta))$
- by solving it, we get $P(w \mid \hat{\theta}) = \frac{c(w, D)}{|D|}$

Which one should we use?

WPM Vs Unigram

Lang pair	setting (source/target)	# vocab.	BLEU
ja→en	Word model (baseline)	80k/80k	28.24
	SentencePiece	8k (shared)	29.55
	SentencePiece w/ pre-tok.	8k (shared)	29.85
	Word/SentencePiece	80k/8k	27.24
	SentencePiece/Word	8k/80k	29.14
en→ja	Word model (baseline)	80k/80k	20.06
	SentencePiece	8k (shared)	21.62
	SentencePiece w/ pre-tok.	8k (shared)	20.86
	Word/SentencePiece	80k/8k	21.41
	SentencePiece/Word	8k/80k	19.94

- ✓ KFTT dataset
  - ✓ Train : dev : test = 440k : 1166 : 1160
  - ✓ English-Japanese translation of Wikipedia articles
- ✓ Model : GNMT(Google's Neural Machine Translation)
- ✓ Pre-Tonkenizer : Moses(en), KyTea(ja)

- **Raw text:** Hello world.
- **Tokenized:** [Hello] [world] [.]

- **Raw text:** [こんにちは世界。] (*Hello world.*)
- **Tokenized:** [こんにちは] [世界] [。]

Which one should we use?

WPM Vs Unigram

Task	Tool	Pre-tok.	time (sec.)	
			Japanese	English
Train	subword-nmt	yes	56.9	54.1
	SentencePiece	yes	10.1	16.8
	subword-nmt	no	528.0	94.7
	SentencePiece	no	217.3	21.8
Seg.	subword-nmt	yes	23.7	28.6
	SentencePiece	yes	8.2	20.3
	subword-nmt	no	216.2	36.1
	SentencePiece	no	5.9	20.3
Pre-tokenizaion KyTea(ja)/Moses(en)			24.6	15.8

- ✓ KFTT dataset
  - ✓ Train : dev : test = 440k : 1166 : 1160
  - ✓ English-Japanese translation of Wikipedia articles
- ✓ Model : GNMT(Google's Neural Machine Translation)
- ✓ Pre-Tonkenizer : Moses(en), KyTea(ja)

- **Raw text:** Hello world.
- **Tokenized:** [Hello] [world] [.]

- **Raw text:** [こんにちは世界。] (*Hello world.*)
- **Tokenized:** [こんにちは] [世界] [。]

Sentence piece used binary heap(O(nlogn))

Which one should we use?

BPE Vs Unigram

```
spm.SentencePieceTrainer.train('--input=botchan.txt --model_prefix=m_bpe --vocab_size=2000 --model_type=bpe')
sp_bpe = spm.SentencePieceProcessor()
sp_bpe.load('m_bpe.model')

print('*** BPE ***')
print(sp_bpe.encode_as_pieces('thisisatesthelloworld'))
print(sp_bpe.nbest_encode_as_pieces('hello world', 5))  # returns an empty list.
```

```
*** BPE ***
['_this', 'is', 'at', 'est', 'he', 'llow', 'or', 'ld']
[]
```

```
spm.SentencePieceTrainer.train('--input=botchan.txt --model_prefix=m_unigram --vocab_size=2000 --model_type=unigram')
sp_unigram = spm.SentencePieceProcessor()
sp_unigram.load('m_unigram.model')

print('*** Unigram ***')
print(sp_unigram.encode_as_pieces('thisisatesthelloworld'))
print(sp_unigram.nbest_encode_as_pieces('thisisatesthelloworld', 5))
```

```
*** Unigram ***
['_this', 'is', 'ate', 's', 'the', 'llow', 'or', 'ld']
[['_this', 'is', 'ate', 's', 'the', 'llow', 'or', 'ld'], ['_this', 'is', 'at', 'es', 'the', 'llow', 'or', 'ld'], ['_this', 'is', 'ate',
's', 'the', 'llow', 'or', 'l', 'd'], ['_this', 'i', 's', 'ate', 's', 'the', 'llow', 'or', 'ld'], ['_this', 'is', 'ate', 'st', 'he', 'llo
w', 'or', 'ld']]
```

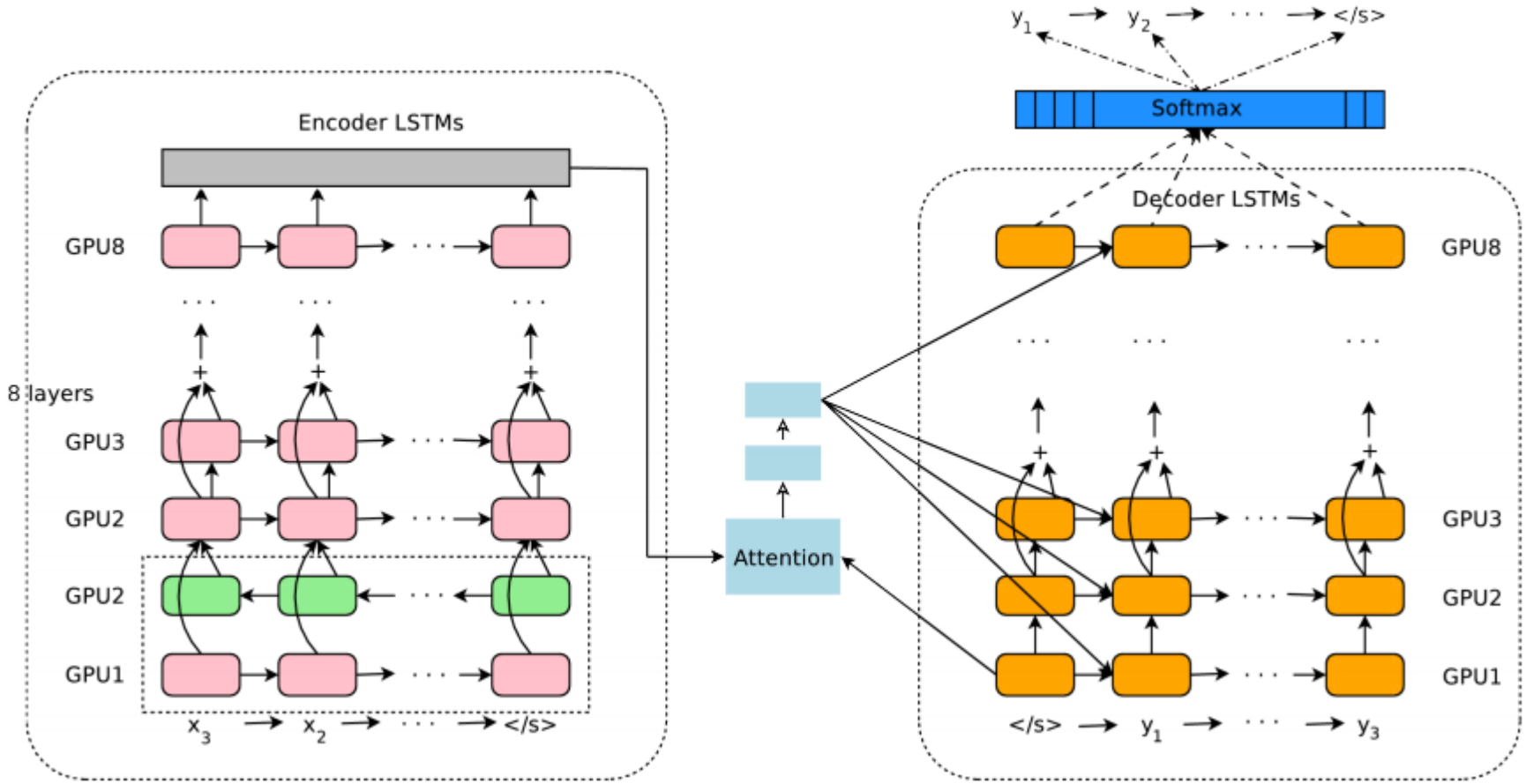
Reference

- Wu, Yonghui, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." *arXiv preprint arXiv:1609.08144* (2016).
- Schuster, Mike, and Kaisuke Nakajima. "Japanese and korean voice search." *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012.
- Kudo, Taku, and John Richardson. "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing." *arXiv preprint arXiv:1808.06226* (2018).
- http://www.phontron.com/kftt/ (KFTT dataset resources)

End

Appendix

GNMT(Google's Neural Machine Translation system)



8 encoder LSTM layers

- 1 bi-directional layer, 7 uni-directional layers

8 decoder LSTM layers