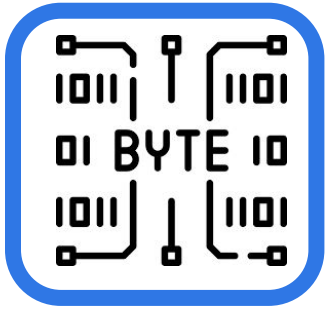
Introduction to NLP for Deep Learning

Ch13 서브워드 토크나이저

Bite Pair Encoding 바이트 페어 인코딩

ch13 analysis 2 page



1. 바이트 페어 인코딩

언어를 더욱 작은 단위로 분해

OOV(out-ofVocab) 문제 해결을 위함

BPE은 기본적으로 연속적으로 가장 많이 등장한 글자의 쌍을 찾아서 하나의 글자로 병합하는 방식을 수행합니다.
주의점은 “aa”등 바이트 쌍을 하나의 바이트 “Z”등으로 변환합니다.

더욱 자세히 요약하자면 글자 단위에서 단어 집합을 만들어냅니다.
예를 들어 가장 많이 등장하는 하나의 유니그램을 하나의 유니그램으로 통합합니다.

예시

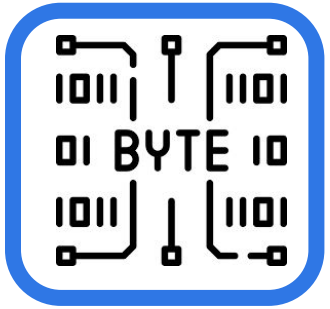
예를 들어 low,lower,등의 단어가 등장했다면 각 단어에대한 단어집합을 만듭니다. 그 후 빈도수가 가장 높은 쌍을 통합합니다. 이 과정을 반복하여 얻는 조합들을 기존의 단어 집합에 추가해줍니다.

vocabulary update!
l, o, w, e, r, n, s, t, i, d, es, est, lo, low, ne, new, newest, wi, wid, widest

예를 들어 low,lower,등의 단어가 등장했다면 그 단어는 low,est로 나뉘는 것입니다.

Bite Pair Encoding 바이트 페어 인코딩

ch13 analysis 3 page



1. 바이트 페어 인코딩

언어를 더욱 작은 단위로 분해

OOV(out-ofVocab) 문제 해결을 위함

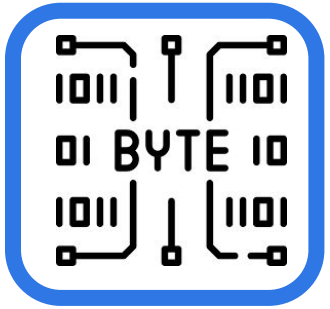
실습

```
def get_stats(dictionary):
    # 유니그램의 pair들의 빈도수를 카운트
    pairs = collections.defaultdict(int)
    for word, freq in dictionary.items():
        symbols = word.split()
        for i in range(len(symbols)-1):
            pairs[symbols[i],symbols[i+1]] += freq
    print('현재 pair들의 빈도수 :', dict(pairs))
    return pairs
dictionary: {}".format(dictionary))
```

```
def merge_dictionary(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

bpe_codes = {}
bpe_codes_reverse = {}

for i in range(num_merges):
    display(Markdown("### Iteration {}".format(i + 1)))
    pairs = get_stats(dictionary)
    best = max(pairs, key=pairs.get)
    dictionary = merge_dictionary(best, dictionary)

    bpe_codes[best] = i
    bpe_codes_reverse[best[0] + best[1]] = best

    print("new merge: {}".format(best))
    print("
```

Bite Pair Encoding 바이트 페어 인코딩

ch13 analysis 4 page



1. 바이트 페어 인코딩

언어를 더욱 작은 단위로 분해

OOV(out-ofVocab) 문제 해결을 위함

WordPiece Tokenizer

WordPiece Tokenizer은 BPE의 변형 알고리즘입니다. 해당 알고리즘은 BPE가 빈도수에 기반하여 가장 많이 등장한 쌍을 병합하는 것과는 달리, 병합되었을 때 코퍼스의 우도(Likelihood)를 가장 높이는 쌍을 병합합니다.

Unigram Language Model Tokenizer

각각의 서브워드들에 대한 손실을 계산합니다.(손실은 코퍼스의 우도가 감소하는 정도)

센텐스피스(SentencePiece)

2. 센텐스 피스

일반적으로 실무에 선택되는 방법

기존의 토크나이저
단어 토큰화 □ 단어 내부 분리
사전 토큰화가 필요하므로 단어 토큰화가 어려운 언어에서는 어렵습니다.

센텐스피스
사전 토큰화 작업 없이 단어 분리 토큰화를 수행하여 언어에 종속되지 않습니다.

```
spm.SentencePieceTrainer.Train('--input=imdb_review.txt
--model_prefix=imdb --vocab_size=5000 --model_type=bpe
--max_sentence_length=9999' )
```

파라미터

input	학습시킬 파일
model_prefix	만들어질 모델 이름
Vocab size	단어 집합 크기
Model type	사용할 모델
Max sentence lengtn	문장 최대 길이
Pad id, pad piece	Pad token id, 값
Unk id, unk piece	Unkown token id, 값
Bos id, bos piece	Begin of sentence token id, 값
Eos id, eos piece	End of sequence token id, 값
User defined symbols	사용자 정의 토큰

```
I didn't at all think of it this way.
['_I', '_didn', "'", 't', '_at', '_all', '_think', '_of', '_it', '_this',
'_way', '.']
[41, 623, 4950, 4926, 138, 169, 378, 30, 58, 73, 413, 4945]

# 단어 내부 분리화 후에 토큰화를 진행함
```

센텐스피스(SentencePiece)

2. 센텐스 피스

일반적으로 실무에 선택되는 방법

```
sp.GetPieceSize()
```

```
5000 # 단어집합의 크기
```

```
sp.IdToPiece(430)
```

```
_think # 정수로부터 맵핑되는 서브워드
```

```
sp.DecodeIds([41, 141, 1364, 1120, 4, 666, 285, 92, 1078, 33, 91])
# 정수 시퀀스로부터 문장 반환
```

```
sp.DecodePieces(['_I', '_have', '_wa', 'ited', '_a', '_long', '_time', '_for', '_someone', '_to', '_film'])
# 서브워드 시퀀스로부터 문장 반환
```

```
sp.PieceToId('_character')
```

```
430 # 서브워드로부터 맵핑되는 정수
```

```
print(sp.encode('I have waited a long time for someone to film', out_type=str))
print(sp.encode('I have waited a long time for someone to film', out_type=int))
['_I', '_have', '_wa', 'ited', '_a', '_long', '_time', '_for', '_someone', '_to', '_film']
[41, 141, 1364, 1120, 4, 666, 285, 92, 1078, 33, 91]
#인자값에 따라 정수, 또는 서브워드 시퀀스로 변환
```

서브워드텍스트인코더(SubwordTextEncoder)

3. 서브 워드 텍스트 인코더

텐서플로우의 서브워드 토크나이저

사용 예

```
tokenizer = tfds.features.text.SubwordTextEncoder.build_from_corpus( train_df['review'], target_vocab_size=2**13)
```

```
tokenizer.subwords
# 토큰화된 서브워드 리스트
```

```
tokenizer.encode(train_df['review'][20])
# 입력데이터에 대해 정수 인코딩 진행
```

```
tokenizer.decode(tokenized_string)
# 정수 인코딩 된 데이터를 문장으로
```

서브워드텍스트인코더(SubwordTextEncoder)

네이버 영화 리뷰 토큰화하기

케라스의 서브워드 토크나이저

데이터 로드

```
urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings_train.txt",
filename="ratings_train.txt")

train_data = pd.read_table('ratings_train.txt')
```

토큰화

이후 NULL을 제거하고 다음과 같이 토큰화 진행한다.

```
tokenizer
= tfds.features.text.SubwordTextEncoder.build_from_corpus(
    train_data['document'], target_vocab_size=2**13)
```

['. ', '..', '영화', '이_', '...', '의_', '는_', '도_', '다', ', ', '을_', '고_', '은_', '가_', '에_', '.. ', '한_', '너무_', '정말_', '를_', '고', '게_', '영화_', '지', '... ', '진짜_', '이', '다_', '요', '만_', '? ', '과_', '나', '가', '서_', '지_', '로_', '으로_', '아', '어', '....', '음', '한', '수_', '와_', '도', '네', '그냥_', '나_', '더_', '왜_', '이런_', '면_', '기', '하고_', '보고_', '하는_', '서', '좀_', '리', '자', '스', '안', '! ', '에서_', '영화를_', '미', 'ㅋㅋ', '네요', '시', '주', '라', '는', '오', '없는_', '에', '해', '사', '!!', '영화는_', '마', '잘_', '수', '영화가_', '만', '본_', '로', '그_', '지만_', '대', '은', '비', '의', '일', '개', '있는_', '없다', '함', '구', '하']

나름 심오한 뜻도 있는 듯. 그냥 학생이 선생과 놀아나는 영화는 절대 아님

[669, 4700, 17, 1749, 8, 96, 131, 1, 48, 2239, 4, 7466, 32, 1274, 2655, 7, 80, 749, 1254]

허깅페이스 토크나이저(Huggingface Tokenizer)

허깅페이스 토크나이저

구글 BERT의 워드 피스 토크나이저

데이터 로드

```
urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings.txt", filename="ratings.txt")
```

토큰화

```
tokenizer = BertWordPieceTokenizer(lowercase=False, trip_accents=False)

#lowercase : 대소문자를 구분 여부. True일 경우 구분하지 않음.
#strip_accents : True일 경우 악센트 제거.
```

```
data_file = 'naver_review.txt';  vocab_size = 30000
limit_alphabet = 6000;    min_frequency = 5

tokenizer.train(files=data_file, vocab_size=vocab_size,
      limit_alphabet=limit_alphabet
,min_frequency=min_frequency)
```

```
# files : 단어 집합을 얻기 위해 학습할 데이터
# vocab_size : 단어 집합의 크기
# limit_alphabet : 병합 전의 초기 토큰의 허용 개수.
# min_frequency : 최소 해당 횟수만큼 등장한 쌍(pair)의 경우에만 병합 대상이 됨
```

```
tokenizer.save_model('./')
df = pd.read_fwf('vocab.txt', header=None)
```

```
#모델 저장 및 로드
```

허깅페이스 토크나이저(Huggingface Tokenizer)

허깅페이스 토크나이저

구글 BERT의 워드 피스 토크나이저

사용 예

```
encoded = tokenizer.encode('커피 한잔의 여유를 즐기다')
print('토큰화 결과 :',encoded.tokens)
print('정수 인코딩 :',encoded.ids)
print('디코딩 :',tokenizer.decode(encoded.ids))
```

```
토큰화 결과 : ['커피', '한잔', '##의', '여유', '##를', '즐기', '##다']
정수 인코딩 : [12825, 25641, 3435, 12696, 3419, 10784, 3260]
디코딩 : 커피 한잔의 여유를 즐기다
```

기타 토크나이저

- BertWordPieceTokenizer : BERT에서 사용된 워드피스 토크나이저CharBPETokenizer : 오리지널 BPE
- ByteLevelBPETokenizer : BPE의 바이트 레벨 버전
- SentencePieceBPETokenizer : 앞서 본 패키지 센텐스피스 (SentencePiece)와 호환되는 BPE 구현체

```
from tokenizers import ByteLevelBPETokenizer, CharBPETokenizer, SentencePieceBPETokenizer tokenizer = SentencePieceBPETokenizer()
tokenizer.train('naver_review.txt', vocab_size=10000, min_frequency=5) encoded = tokenizer.encode("이 영화는 정말 재미있습니다.") print(encoded.tokens)
```