딥러닝을 이용한 자연어 처리 입문

13. 서브워드 토크나이저 (Subword Tokenizer)

집현전 3기 초급 3조 – 김보겸, 이동형, 김성범

0) 목차

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

기계에 아무리 많은 단어를 학습시켜도 세상에 존재하는 모든 단어를 학습시키는 것은 불가능

-> 모르는 단어가 등장했을 때 문제를 해결하기 어려운 상황 (Out Of Vocabulary 문제) 발생

서브워드 분리 (Subword Segmentation) 작업

- 하나의 단어를 더욱 작은 단위의 여러 서브워드로 분리하는 전처리 작업
  - Ex) birthday = birth + day / 매운맛 = 매운 + 맛
- OOV나 희귀 단어, 신조어 문제를 완화시킬 수 있음

⇨ **가장 대표적인 서브워드 분리 알고리즘인 BPE (Byte Pair Encoding)에 대해 알아봅시다**

Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.1) BPE (Byte Pair Encoding)

- 1994년 데이터 압축 알고리즘으로 제안되었음, 이후 자연어 처리의 서브워드 분리 알고리즘으로 응용
- 연속적으로 가장 많이 등장한 글자의 쌍을 찾아서 하나의 글자로 병합하는 방식

```
aaabdaaabac
```

```
ZabdZabac
Z=aa
```

```
ZYdZYac
Y=ab
Z=aa
```

```
XdXac
X=ZY
Y=ab
Z=aa
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.2) 자연어 처리에서의 BPE

- 자연어 처리에서는 BPE를 서브워드 분리 알고리즘으로서 사용
- 글자(character) 단위에서 점차적으로 단어 집합(vocabulary)를 만들어내는 bottom up 방식의 접근
- 훈련 데이터에 있는 모든 글자 또는 유니코드 단위로 분해하여 단어 집합 생성 -> 가장 많이 등장하는 유니그램을 하나의 유니그램으로 통합

```
# dictionary
l o w : 5,  l o w e r : 2,  n e w e s t : 6,  w i d e s t : 3
```

```
# vocabulary
l, o, w, e, r, n, w, s, t, i, d
```

1회 - 딕셔너리를 참고로 하였을 때 빈도수가 9로 가장 높은 (e, s)의 쌍을 es로 통합합니다.

```
# dictionary update!
l o w : 5,
l o w e r : 2,
n e w es t : 6,
w i d es t : 3
```

```
# vocabulary update!
l, o, w, e, r, n, w, s, t, i, d, es
```

2회 - 빈도수가 9로 가장 높은 (es, t)의 쌍을 est로 통합합니다.

```
# dictionary update!
l o w : 5,
l o w e r : 2,
n e w est : 6,
w i d est : 3
```

```
# vocabulary update!
l, o, w, e, r, n, w, s, t, i, d, es, est
```

총 10회 반복

```
# dictionary update!
low : 5,
low e r : 2,
newest : 6,
widest : 3
```

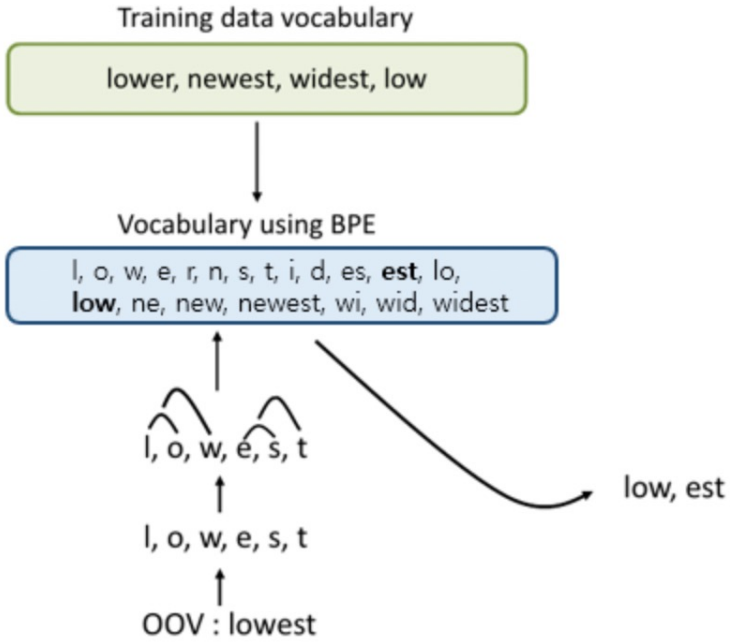
```
# vocabulary update!
l, o, w, e, r, n, w, s, t, i, d, es, est, lo, low, ne, new, newest, wi, wid, widest
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.2) 자연어 처리에서의 BPE

- 새로운 단어가 입력되었을 때: 새로 들어오려는 단어를 여러 서브워드로 나누고, 그것들을 Vocabulary 안에서 있는지 찾아내는 원리



Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.3) 코드 실습하기

1. 필요한 도구 임포트

```
import re, collections
from IPython.display import display, Markdown, Latex
```

2. BPE 수행 횟수, Dictionary 설정

```
num_merges = 10 # BPE를 몇 회 수행할 것인지를 설정

dictionary = {     # BPE 알고리즘에서 사용할 dictionary, </w>는 단어의 맨 끝에 붙이는 특수문자, 각 단어는 글자 단위로 분할
    '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
}
```

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.3) 코드 실습하기

- 가장 빈도수가 높은 유니그램의 쌍을 하나의 유니그램으로 통합하는 과정을 num_merges회 (10회) 반복하는 코드

3. 필요한 함수 생성

```
# 유니그램 pair들의 빈도수를 카운트하는 함수

def get_stats(dictionary):
    pairs = collections.defaultdict(int) # 모든 키에 대해서 값이 없는 경우 자동으로 생성자의 인자로 넘어온 함수를 호출하여 그 결과값으로 설정하는 함수 사용
    for word, freq in dictionary.items():  # word에 단어, freq에 빈도 할당
        symbols = word.split()  # 띄어쓰기 단위로 분할 ex: l, o, w, </w>
        for i in range(len(symbols)-1):
            pairs[symbols[i], symbols[i+1]] += freq   # 문자와 그 다음 문자 사이를 하나의 페어로 묶어주고, freq만큼 빈도수를 늘려줌 Ex. l,o 조합을 5만큼 늘려줌
    print(f'현재 pair들의 빈도수 :{dict(pairs)}')
    return pairs
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.3) 코드 실습하기

- 가장 빈도수가 높은 유니그램의 쌍을 하나의 유니그램으로 통합하는 과정을 num_merges회 (10회) 반복하는 코드

3. 필요한 함수 생성

```
# pair를 반영하여 딕셔너리를 업데이트할 수 있는 함수

def merge_dictionary(pair, v_in):
    v_out = {}
    bigram = re.escape(' '.join(pair)) # 각각 쪼개져서 존재하는 글자들을 bigram 형태로 변환, </w>를 위해서 이스케이프 처리도 해줌
    p = re.compile(r'(?<!\S)'+ bigram + r'(?!\S)')
    for word in v_in:
        w_out = p.sub(''.join(pair),word) # 새로 만들어진 페어(lo)들을 원래 딕셔너리에 반영하기 (교체)
        v_out[w_out] = v_in[word] # 새로 만든 딕셔너리의 value 값이나 교체 이전의 value 값은 어짜피 똑같음!
    return v_out    # pair가 반영된 딕셔너리를 반환
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.3) 코드 실습하기

- 가장 빈도수가 높은 유니그램의 쌍을 하나의 유니그램으로 통합하는 과정을 num_merges회 (10회) 반복하는 코드

4. 함수들을 조합하여 BPE 알고리즘 구현

```
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) # pair의 값(비도수)이 가장 큰 pair를 best에 할당
    dictionary = merge_dictionary(best, dictionary)  # best에 할당된 pair를 반영하도록 dictionary를 업데이트

    bpe_codes[best] = i  # best에 할당된 pair들을 순서대로 저장
    bpe_codes_reverse[best[0] + best[1]] = best  # 이 친구는 왜 쓰는지 모르겠습니다...

    print(f"new merge: {best}")
    print(f"dictionary: {dictionary}")
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.3) 코드 실습하기

- 가장 빈도수가 높은 유니그램의 쌍을 하나의 유니그램으로 통합하는 과정을 num_merges회 (10회) 반복하는 코드

5. 결과

```
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) # pair의 값(빈도수)이 가장 큰 pair를 best에 할당
    dictionary = merge_dictionary(best, dictionary)  # best에 할당된 pair를 반영하도록 dictionary를 업데이트

    bpe_codes[best] = i  # best에 할당된 pair들을 순서대로 저장
    bpe_codes_reverse[best[0] + best[1]] = best  # 이 친구는 왜 쓰는지 모르겠습니다...

    print(f"new merge: {best}")
    print(f"dictionary: {dictionary}")
```

```
dictionary = {'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
              }
```

```
Iteration 1
현재 pair들의 빈도수 : {('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}
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.3) 코드 실습하기

- 가장 빈도수가 높은 유니그램의 쌍을 하나의 유니그램으로 통합하는 과정을 num_merges회 (10회) 반복하는 코드

5. 결과

```
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) # pair의 값(비도수)이 가장 큰 pair를 best에 할당
    dictionary = merge_dictionary(best, dictionary)  # best에 할당된 pair를 반영하도록 dictionary를 업데이트

    bpe_codes[best] = i  # best에 할당된 pair들을 순서대로 저장
    bpe_codes_reverse[best[0] + best[1]] = best  # 이 친구는 왜 쓰는지 모르겠습니다...

    print(f"new merge: {best}")
    print(f"dictionary: {dictionary}")
```

```
print(bpe_codes)
```

```
{('e', 's'): 0, ('es', 't'): 1, ('est', '</w>'): 2, ('l', 'o'): 3, ('lo', 'w'): 4, ('n', 'e'): 5, ('ne',
'w'): 6, ('new', 'est</w>'): 7, ('low', '</w>'): 8, ('w', 'i'): 9}
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.4) OOV 대처하기

- 1.3에서 Vocabulary를 만들었다면, 1.4에서는 1.3에서 만든 vocab을 바탕으로 새로 입력되는 단어들을 인코딩하는 코드를 짜봅니다

1. 필요한 함수 만들기

```
# 단어들을 바탕으로 pair를 만들어내는 함수

def get_pairs(word):
    pairs = set()
    prev_char = word[0]
    for char in word[1:]:
        pairs.add((prev_char, char))
        prev_char = char
        print(f'pair:{pairs}, prev_char: {prev_char}')
    return pairs
```

```
word split into characters: ('l', 'o', 'k', 'i', '')
pair:{('l', 'o')}, prev_char: o
pair:{('l', 'o'), ('o', 'k')}, prev_char: k
pair:{('l', 'o'), ('k', 'i'), ('o', 'k')}, prev_char: i
pair:{('l', 'o'), ('k', 'i'), ('i', '</w>'), ('o', 'k')}, prev_char: </w>
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.4) OOV 대처하기

- 1.3에서 Vocabulary를 만들었다면, 1.4에서는 1.3에서 만든 vocab을 바탕으로 새로 입력되는 단어들을 인코딩하는 코드를 짜봅니다

1. 필요한 함수 만들기

```
# 만들어진 pair를 바탕으로 BPE 알고리즘에 따라 서브워드 단어집합을 생성하는 함수

def encode(orig):
    word = tuple(orig) + ('</w>',) # ex. tuple('loki') -> ('l','o','k','i','')
    display(Markdown("__word split into characters:__ <tt>{}</tt>".format(word)))

    pairs = get_pairs(word)  # 위에서 만든 get_pair 함수를 통해 pair 생성하기
    if not pairs:
        return orig
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.4) OOV 대처하기

- 1.3에서 Vocabulary를 만들었다면, 1.4에서는 1.3에서 만든 vocab을 바탕으로 새로 입력되는 단어들을 인코딩하는 코드를 짜봅니다

1. 필요한 함수 만들기

```
iteration = 0
while True:
    iteration += 1
    display(Markdown("__Iteration {}:__".format(iteration)))

    print(f"bigrams in the word: {pairs}")
    # 전에 생성한 bpe_codes안에 새로 생성한 pair가 존재하는지보고 (= subword vocab 안에 있는지 보고)
    # bpe_codes의 value 중에 가장 낮은 pair로 bigram 생성 (= 순서대로 검증)
    bigram = min(pairs, key = lambda pair: bpe_codes.get(pair, float('inf')))
    print(f"candidate for merging: {bigram}")
    if bigram not in bpe_codes:        # 만약에 bigram이 bpe_code에 없으면 중단
        display(Markdown("__Candidate not in BPE merges, algorithm stops.__"))
        break
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.4) OOV 대처하기

- 1.3에서 Vocabulary를 만들었다면, 1.4에서는 1.3에서 만든 vocab을 바탕으로 새로 입력되는 단어들을 인코딩하는 코드를 짜봅니다

1. 필요한 함수 만들기

```
first, second = bigram              # bigram 변수는 ('l','o') 등의 형태로 존재 > 언패킹
new_word = []
i = 0
while i < len(word):                # word = ('l', 'o', 'k', 'i', '')
    try:
        j = word.index(first, i)    # tuple.index(value, start, stop)
        new_word.extend(word[i:j])
        i = j
    except:                         # index를 못찾았을 때
        new_word.extend(word[i:])
        break

    if word[i] == first and i < len(word)-1 and word[i+1] == second:
        new_word.append(first+second)
        i += 2
    else:
        new_word.append(word[i])
        i += 1
new_word = tuple(new_word)
print(f"new_word: {new_word}")
word = new_word                     # 기존 word를 new_word로 교체
print(f"word after merging: {word}")
if len(word) == 1:
    break
else:
    pairs = get_pairs(word)
```

```
# 특별 토큰인 </w>는 출력하지 않는다.
if word[-1] == '</w>':
    word = word[:-1]
elif word[-1].endswith('</w>'):
    word = word[:-1] + (word[-1].replace('</w>',''),)

return word
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

1.4) OOV 대처하기

- 1.3에서 Vocabulary를 만들었다면, 1.4에서는 1.3에서 만든 vocab을 바탕으로 새로 입력되는 단어들을 인코딩하는 코드를 짜봅니다

2. 제작한 함수로 인코딩 해보기

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

```
encode('loki')
```

word split into characters: ('l', 'o', 'k', 'i', '')

Iteration 1:

```
bigrams in the word: {('l', 'o'), ('k', 'i'), ('i', '</w>'), ('o', 'k')}
candidate for merging: ('l', 'o')
new_word: ('lo', 'k', 'i', '</w>')
word after merging: ('lo', 'k', 'i', '</w>')
```

Iteration 2:

```
bigrams in the word: {('k', 'i'), ('lo', 'k'), ('i', '</w>')}
candidate for merging: ('k', 'i')
```

Candidate not in BPE merges, algorithm stops.

```
('lo', 'k', 'i')
```

```
encode('highing')
```

word split into characters: ('h', 'i', 'g', 'h', 'i', 'n', 'g', '')

Iteration 1:

```
bigrams in the word: {('g', 'h'), ('i', 'g'), ('n', 'g'), ('h', 'i'), ('i', 'n'), ('g', '</w>')}
candidate for merging: ('g', 'h')
```

Candidate not in BPE merges, algorithm stops.

```
('h', 'i', 'g', 'h', 'i', 'n', 'g')
```


Enjoy your stylish business and campus life with BIZCAM

1) 바이트 페어 인코딩 (Byte Pair Encoding, BPE)

WordPiece Tokenizer

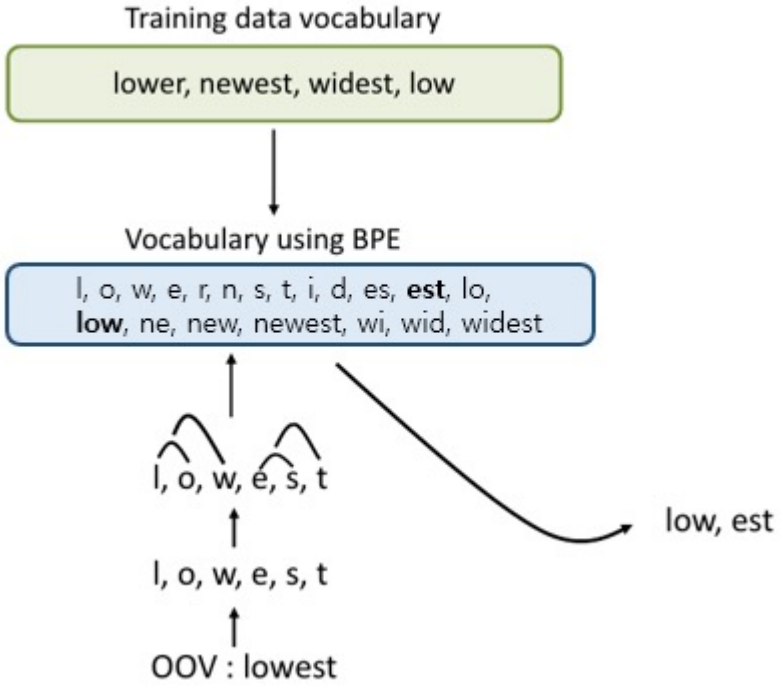
- BPE의 변형 알고리즘
- BPE가 빈도수에 기반하여 가장 많이 등장한 쌍을 병합
- WordPiece는 코퍼스의 likelihood를 가장 높이는 쌍을 병합
- **수행하기 이전의 문장:** Jet makers feud over seat width with big orders at stake

  WordPiece Tokenizer를 수행한 결과(wordpieces): _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

Unigram Language Model Tokenizer

- 각각의 서브워드들에 대해서 손실(loss) 계산
  - 손실이란 해당 서브워드가 단어 집합에서 제거되었을 경우, 코퍼스의 likelihood가 감소하는 정도
- 측정된 서브워드들을 손실의 정도로 정렬, 최악의 영향을 주는 10~20%의 토큰 제거

2) 센텐스피스(SentencePiece)



BPE를 포함하여 기타 서브워드 토크나이징 알고리즘들을 내장한 센텐스피스(SentencePiece) 소개

Enjoy your stylish business and campus life with BIZCAM

2) 센텐스피스(SentencePiece)

내부 단어 분리를 위한 유용한 패키지로 구글의 센텐스피스(Sentencepiece)

구글은 BPE 알고리즘과 Unigram Language Model Tokenizer를 구현한 센텐스피스

센텐스피스는 토큰화 작업(Pretokenization)없이 raw data에 바로 적용 가능하도록 구현

센텐스피스는 사전 토큰화 작업없이 단어 분리 토큰화를 수행하므로 언어에 종속되지 않음.

즉, 센텐스피스는 미등록 어휘(Out of Vocabulary, OOV)에 대응한 어휘 모델을 생성하기 위해 입력 문장을 받아 언어에 의존하지 않은 서브 워드 모델을 학습

Enjoy your stylish business and campus life with BIZCAM

2) 센텐스피스(SentencePiece)

2.1) IMDB 리뷰 토큰화하기

1. 라이브러리 설정

```
import sentencepiece as spm # pip install sentencepiece
import pandas as pd
import urllib.request
import csv
```

2. IMDB 리뷰 데이터 다운 및 DataFrame 저장

```
urllib.request.urlretrieve("https://raw.githubusercontent.com/LawrenceDuan/IMDb-Review-Analysis/master/IMDb_Reviews.csv
                           filename="IMDb_Reviews.csv")
train_df = pd.read_csv('IMDb_Reviews.csv')
train_df['review']
```

```
0        My family and I normally do not watch local mo...
1        Believe it or not, this was at one time the wo...
2        After some internet surfing, I found the "Home...
3        One of the most unheralded great works of anim...
4        It was the Sixties, and anyone with long hair ...
                               ...
49995    the people who came up with this are SICK AND ...
49996    The script is so so laughable... this in turn,...
49997    "So there's this bride, you see, and she gets ...
49998    Your mind will not be satisfied by this no□bud...
49999    The chaser's war on everything is a weekly sho...
Name: review, Length: 50000, dtype: object
```

```
print('리뷰 개수 :',len(train_df)) # 리뷰 개수 출력
```

```
리뷰 개수 : 50000
```

5만개 데이터 샘플 확인

2) 센텐스피스(SentencePiece)

2.1) IMDB 리뷰 토큰화하기

3. DataFrame을 txt 파일로 저장

```
# 센텐스피스의 입력으로 사용하기 위해서 데이터프레임을 txt 파일로 저장

with open('imdb_review.txt', 'w', encoding='utf8') as f:
    f.write('₩n'.join(train_df['review']))
```

4. 센텐스피스로 단어 집합과 각 단어 고유 정수 부여

```
# 센텐스피스로 단어 집합과 각 단어에 고유한 정수를 부여

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

- input : 학습시킬 파일
- model_prefix : 만들어질 모델 이름
- vocab_size : 단어 집합의 크기
- model_type : 사용할 모델 (unigram(default), bpe, char, word)
- max_sentence_length: 문장의 최대 길이

2) 센텐스피스(SentencePiece)

2.1) IMDB 리뷰 토큰화하기

5. Vocab 생성 후 DataFrame으로 저장

```
# vocab 생성이 완료되면 imdb.model, imdb.vocab 파일 두개가 생성
# 단어 집합의 크기를 확인하기 위해 vocab 파일을 데이터프레임에 저장

vocab_list = pd.read_csv('imdb.vocab', sep='₩t', header=None, quoting=csv.QUOTE_NONE)
vocab_list.sample(10)
```

	0	1
2679	_innoc	-2676
507	_little	-504
2859	_funn	-2856
2364	_mix	-2361
1272	_couple	-1269
3399	riage	-3396
1666	_shoot	-1663
1610	_present	-1607
4290	_tears	-4287
2437	_IMD	-2434

```
# 단어 집합의 크기는 5,000개

len(vocab_list)
```

```
5000
```

```
# model 파일을 로드

sp = spm.SentencePieceProcessor()
vocab_file = "imdb.model"
sp.load(vocab_file)
```

```
True
```


Enjoy your stylish business and campus life with BIZCAM

2) 센텐스피스(SentencePiece)

2.1) IMDB 리뷰 토큰화하기

6. 인코더 디코더 변환 확인

```
lines = [
  "I didn't at all think of it this way.",
  "I have waited a long time for someone to film"
]
for line in lines:
  print(line)
  print(sp.encode_as_pieces(line))
  print(sp.encode_as_ids(line))
  print()
```

```
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]

I have waited a long time for someone to film
['_I', '_have', '_wa', 'ited', '_a', '_long', '_time', '_for', '_someone', '_to', '_film']
[41, 141, 1364, 1120, 4, 666, 285, 92, 1078, 33, 91]
```

Encode_as_pieces : 문장을 입력하면 서브 워드 시퀀스로 변환
encode_as_ids : 문장을 입력하면 정수 시퀀스로 변환

Enjoy your stylish business and campus life with BIZCAM

2) 센텐스피스(SentencePiece)

2.1) IMDB 리뷰 토큰화하기

```
# GetPieceSize() : 단어 집합의 크기를 확인

sp.GetPieceSize()
```

```
5000
```

```
# idToPiece : 정수로부터 맵핑되는 서브 워드로 변환

sp.IdToPiece(430)
```

```
'_character'
```

```
# PieceToId : 서브워드로부터 맵핑되는 정수로 변환

sp.PieceToId('_character')
```

```
430
```

```
# DecodeIds : 정수 시퀀스로부터 문장으로 변환

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

```
'I have waited a long time for someone to film'
```

```
# DecodePieces : 서브워드 시퀀스로부터 문장으로 변환

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

```
'I have waited a long time for someone to film'
```

```
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]
```


Enjoy your stylish business and campus life with BIZCAM

2) 센텐스피스(SentencePiece)

2.2) 네이버 영화 리뷰 토큰화

1. 네이버 영화 리뷰 데이터 다운 및 DataFrame 저장

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

```
('ratings.txt', <http.client.HTTPMessage at 0x7effe8d233d0>)
```

```
naver_df = pd.read_table('ratings.txt')
naver_df[:5]
```

	id	document	label
0	8112052	어릴때보고 지금다시봐도 재밌어요ㅋㅋ	1
1	8132799	디자인을 배우는 학생으로, 외국디자이너와 그들이 일군 전통을 통해 발전해가는 문화산...	1
2	4655635	폴리스스토리 시리즈는 1부터 뉴까지 버릴께 하나도 없음.. 최고.	1
3	9251303	와.. 연기가 진짜 개쩔구나.. 지루할거라고 생각했는데 몰입해서 봤다.. 그래 이런...	1
4	10067386	안개 자욱한 밤하늘에 떠 있는 초승달 같은 영화.	1

```
print('리뷰 개수 :',len(naver_df)) # 리뷰 개수 출력
```

```
리뷰 개수 : 200000
```

```
# null 값 제거

print(naver_df.isnull().values.any())
```

```
True
```

```
naver_df = naver_df.dropna(how = 'any') # Null 값이 존재하는 행 제거
print(naver_df.isnull().values.any()) # Null 값이 존재하는지 확인
```

```
False
```

```
print('리뷰 개수 :',len(naver_df)) # 리뷰 개수 출력
```

```
리뷰 개수 : 199992
```


Enjoy your stylish business and campus life with BIZCAM

2) 센텐스피스(SentencePiece)

2.2) 네이버 영화 리뷰 토큰화

2. 센텐스피스로 단어 집합과 각 단어 고유 정수 부여 후 DataFrame으로 저장

```
# 199,992개의 샘플을 naver_review.txt 파일에 저장한 후에 센텐스피스를 통해 단어 집합을 생성

with open('naver_review.txt', 'w', encoding='utf8') as f:
    f.write('₩n'.join(naver_df['document']))
```

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

```
vocab_list = pd.read_csv('naver.vocab', sep='₩t', header=None, quoting=csv.QUOTE_NONE)
vocab_list[:10]
```

	0	1
0	<unk>	0
1	<s>	0
2	</s>	0
3	..	0
4	영화	-1
5	_영화	-2
6	_이	-3
7	_아	-4
8	...	-5
9	_그	-6

```
sp = spm.SentencePieceProcessor()
vocab_file = "naver.model"
sp.load(vocab_file)
```

```
True
```

vocab_list.sample(10)

	0	1
2389	_심심	-2386
3741	믿	-3738
1167	_영성	-1164
2294	로서	-2291
2882	속에서	-2879
4684	₩	-4681
1860	_아빠	-1857
2894	_쓸데	-2891
4624	쌤	-4621
263	_멋	-260

Enjoy your stylish business and campus life with BIZCAM

2) 센텐스피스(SentencePiece)

2.2) 네이버 영화 리뷰 토큰화

3. 인코더 디코더 변환 확인

```
lines = [
  "뭐 이딴 것도 영화냐.",
  "진짜 최고의 영화입니다 ㅋㅋ",
]
for line in lines:
  print(line)
  print(sp.encode_as_pieces(line))
  print(sp.encode_as_ids(line))
  print()
```

```
뭐 이딴 것도 영화냐.
['_뭐', '_이딴', '_것도', '_영화냐', '.']
[132, 966, 1296, 2590, 3276]

진짜 최고의 영화입니다 ㅋㅋ
['_진짜', '_최고의', '_영화입니다', '_ㅋㅋ']
[54, 200, 821, 85]
```

```
sp.GetPieceSize()
```

```
5000
```

```
sp.IdToPiece(4)
```

```
'영화'
```

```
sp.PieceToId('영화')
```

```
4
```

```
sp.DecodeIds([54, 200, 821, 85])
```

```
'진짜 최고의 영화입니다 ㅋㅋ'
```


Enjoy your stylish business and campus life with BIZCAM

3) 서브워드텍스트인코더(SubwordTextEncoder)

SubwordTextEncoder는 텐서플로우를 통해 사용할 수 있는 서브워드 토크나이저

BPE와 유사한 알고리즘인 Wordpiece Model을 채택하였으며,

패키지를 통해 쉽게 단어들을 서브워드들로 분리

SubwordTextEncoder를 통해서 IMDB 영화 리뷰 데이터와 네이버 영화 리뷰 데이터에 토큰화 작업 실습

Enjoy your stylish business and campus life with BIZCAM

3) 서브워드텍스트인코더(SubwordTextEncoder)

3.1) IMDB 리뷰 토큰화하기

1. 라이브러리 설정 및 IMDB 리뷰 다운로드

```
import tensorflow_datasets as tfds # Tensorflow 2.3+ 버전에서는 tfds.features.text 대신 tfds.deprecated.text라고 작성
import urllib.request
import pandas as pd
```

```
urllib.request.urlretrieve("https://raw.githubusercontent.com/LawrenceDuan/IMDb-Review-Analysis/master/IMDb_Reviews.csv", filename="IMDb_Reviews.csv")
```

```
('IMDb_Reviews.csv', <http.client.HTTPMessage at 0x7fb67a8518d0>)
```

```
train_df = pd.read_csv('IMDb_Reviews.csv')
```

```
train_df['review']
```

```
0        My family and I normally do not watch local mo...
1        Believe it or not, this was at one time the wo...
2        After some internet surfing, I found the "Home...
3        One of the most unheralded great works of anim...
4        It was the Sixties, and anyone with long hair ...
                               ...
49995    the people who came up with this are SICK AND ...
49996    The script is so so laughable... this in turn,...
49997    "So there's this bride, you see, and she gets ...
49998    Your mind will not be satisfied by this no□bud...
49999    The chaser's war on everything is a weekly sho...
Name: review, Length: 50000, dtype: object
```


Enjoy your stylish business and campus life with BIZCAM

3) 서브워드텍스트인코더(SubwordTextEncoder)

3.1) IMDB 리뷰 토큰화하기

2. 서브워드들로 이루어진 단어 집합(Vocabulary) 생성 후 각 서브워드에 고유 정수 부여

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

```
print(tokenizer.subwords[:100])
```

```
['the_', ', ', '. ', 'a_', 'and_', 'of_', 'to_', 's_', 'is_', 'br', 'in_', 'I_', 'that_', 'this_', 'it_', ' /><', ' />',
```

3. 20번 인덱스 샘플 출력 후 정수 인코딩 결과 비교

```
# 임의로 선택한 21번째 샘플을 출력해보고, 정수 인코딩을 수행한 결과와 비교

print(train_df['review'][20])
```

```
Pretty bad PRC cheapie which I rarely bother to watch over again, and it's no wonder -- it's slow and creaky and dull as a butter knife. Mad doctor George
```

```
print('Tokenized sample question: {}'.format(tokenizer.encode(train_df['review'][20])))
```

```
Tokenized sample question: [1590, 4162, 132, 7107, 1892, 2983, 578, 76, 12, 4632, 3422, 7, 160, 175, 372, 2, 5, 39, 8051, 8, 84, 2652, 497, 39, 8051, 8, 1
```


Enjoy your stylish business and campus life with BIZCAM

3) 서브워드텍스트인코더(SubwordTextEncoder)

3.1) IMDB 리뷰 토큰화하기

4. 짧은 문장에 대해 정수 인코딩 결과 및 디코딩 진행

```
# train_df에 존재하는 문장 중 일부를 발췌
sample_string = "It's mind-blowing to me that this film was even made."

# 인코딩한 결과를 tokenized_string에 저장
tokenized_string = tokenizer.encode(sample_string)
print ('정수 인코딩 후의 문장 {}'.format(tokenized_string))

# 이를 다시 디코딩
original_string = tokenizer.decode(tokenized_string)
print ('기존 문장: {}'.format(original_string))
```

```
정수 인코딩 후의 문장 [137, 8051, 8, 910, 8057, 2169, 36, 7, 103, 13, 14, 32, 18, 79, 681, 8058]
기존 문장: It's mind-blowing to me that this film was even made.
```

```
print('단어 집합의 크기(Vocab size) :', tokenizer.vocab_size)
```

```
단어 집합의 크기(Vocab size) : 8268
```

```
for ts in tokenized_string:
  print ('{} ----> {}'.format(ts, tokenizer.decode([ts])))
```

```
137 ----> It
8051 ----> '
8 ----> s
910 ----> mind
8057 ----> -
2169 ----> blow
36 ----> ing
7 ----> to
103 ----> me
13 ----> that
14 ----> this
32 ----> film
18 ----> was
79 ----> even
681 ----> made
8058 ----> .
```


Enjoy your stylish business and campus life with BIZCAM

3) 서브워드텍스트인코더(SubwordTextEncoder)

3.1) IMDB 리뷰 토큰화하기

5. Even 단어에 임의의 xyz 3개 글자 추가 실험

```
# 앞서 실습한 문장에 even 뒤에 임의로 xyz 추가
sample_string = "It's mind-blowing to me that this film was evenxyz made."

# 인코딩한 결과를 tokenized_string에 저장
tokenized_string = tokenizer.encode(sample_string)
print ('정수 인코딩 후의 문장 {}'.format(tokenized_string))

# 이를 다시 디코딩
original_string = tokenizer.decode(tokenized_string)
print ('기존 문장: {}'.format(original_string))
```

```
정수 인코딩 후의 문장 [137, 8051, 8, 910, 8057, 2169, 36, 7, 103, 13, 14, 32, 18, 7974, 8132, 8133, 997, 681, 8058]
기존 문장: It's mind-blowing to me that this film was evenxyz made.
```

```
for ts in tokenized_string:
  print ('{} ----> {}'.format(ts, tokenizer.decode([ts])))
```

```
137 ----> It
8051 ----> '
8 ----> s
910 ----> mind
8057 ----> -
2169 ----> blow
36 ----> ing
7 ----> to
103 ----> me
13 ----> that
14 ----> this
32 ----> film
18 ----> was
7974 ----> even
8132 ----> x
8133 ----> y
997 ----> z
681 ----> made
8058 ----> .
```


Enjoy your stylish business and campus life with BIZCAM

3) 서브워드텍스트인코더(SubwordTextEncoder)

3.2) 네이버 영화 리뷰 토큰화

1. 라이브러리 설정 및 데이터 프레임 저장

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

```
('ratings_train.txt', <http.client.HTTPMessage at 0x7fb676878410>)
```

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

```
# null 값 제거

print(train_data.isnull().sum())
```

```
id          0
document    5
label       0
dtype: int64
```

```
train_data = train_data.dropna(how = 'any') # Null 값이 존재하는 행 제거
print(train_data.isnull().values.any()) # Null 값이 존재하는지 확인
```

```
False
```


Enjoy your stylish business and campus life with BIZCAM

3) 서브워드텍스트인코더(SubwordTextEncoder)

3.2) 네이버 영화 리뷰 토큰화

2. 서브워드들로 이루어진 단어 집합(Vocabulary) 생성 후 각 서브워드에 고유 정수 부여

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

```
print(tokenizer.subwords[:100])
```

```
['. ', ', ', '영화', '이_', '...', '익_', '는_', '도_', '다', ', ', '을_', '고_', '은_', '가_', '에_', '.. ', '한_', '너무_',
```

```
print('Tokenized sample question: {}'.format(tokenizer.encode(train_data['document'][20])))
```

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


Enjoy your stylish business and campus life with BIZCAM

3) 서브워드텍스트인코더(SubwordTextEncoder)

3.2) 네이버 영화 리뷰 토큰화

3. 20번 인덱스 샘플 출력 후 정수 인코딩 결과 비교, 21번 인덱스 인코딩 및 디코딩

```
print('Tokenized sample question: {}',format(tokenizer.encode(train_data['document'][20])))
```

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

```
sample_string = train_data['document'][21]

# 인코딩한 결과를 tokenized_string에 저장
tokenized_string = tokenizer.encode(sample_string)
print ('정수 인코딩 후의 문장 {}',format(tokenized_string))

# 이를 다시 디코딩
original_string = tokenizer.decode(tokenized_string)
print ('기존 문장: {}',format(original_string))
```

```
정수 인코딩 후의 문장 [570, 892, 36, 584, 159, 7091, 201]
기존 문장: 보면서 웃지 않는 건 불가능하다
```

```
for ts in tokenized_string:
  print ('{} ----> {}'.format(ts, tokenizer.decode([ts])))
```

```
570 ----> 보면서
892 ----> 웃
36 ----> 지
584 ----> 않는
159 ----> 건
7091 ----> 불가능
201 ----> 하다
```


Enjoy your stylish business and campus life with BIZCAM

4) 허깅페이스 토크나이저 (Huggingface Tokenizer)

4.1) BERT의 워드피스 토크나이저(BertWordPieceTokenizer)

1. 필요한 도구 설치 및 임포트

```
[ ] pip install tokenizers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tokenizers in /usr/local/lib/python3.7/dist-packages (0.12.1)
```

```
[ ] import pandas as pd
    import urllib.request
    from tokenizers import BertWordPieceTokenizer
```

```
('ratings.txt', <http.client.HTTPMessage at 0x7f0cacacea50>)
```

2. 파일 생성 후 결측치 제거, 저장

```
# 결측치 제거 후, 실질적인 리뷰 데이터인 document 열에 대해서 naver_review.txt 파일을 새로 만들어 저장
urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings.txt", filename="ratings.txt")

naver_df = pd.read_table('ratings.txt')
naver_df = naver_df.dropna(how='any')
with open('naver_review.txt', 'w', encoding='utf-8') as f:
  f.write('\n'.join(naver_df['document']))
```


Enjoy your stylish business and campus life with BIZCAM

4) 허깅페이스 토크나이저 (Huggingface Tokenizer)

4.1) BERT의 워드피스 토크나이저(BertWordPieceTokenizer)

3. Tokenizer 생성 및 리뷰 데이터 학습

```
# BerWordPieceTokenizer 설정
# lowercase: 대소문자 구분 여부. True일 경우 구분하지 않음 / strip_accents: True일 경우 악센트 제거 (ex. é → e, ô → o)

tokenizer = BertWordPieceTokenizer(lowercase=False, strip_accents=False)    # 교안에 'trip_accents가 아닌 trip_accents로 오타 나있음!
```

```
# 리뷰 데이터를 학습하여 단어 집합 얻기
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 병합의 대상이 됨

Enjoy your stylish business and campus life with BIZCAM

4) 허깅페이스 토크나이저 (Huggingface Tokenizer)

4.1) BERT의 워드피스 토크나이저(BertWordPieceTokenizer)

3. Tokenizer 생성 및 리뷰 데이터 학습 - 결과

```
# vocab 저장 (현재 경로)
tokenizer.save_model('./')
```

```
['./vocab.txt']
```

```
# vocab 로드
df = pd.read_fwf('vocab.txt', header=None)
df
```

	0
0	[PAD]
1	[UNK]
2	[CLS]
3	[SEP]
4	[MASK]
...	...
29995	말라는
29996	말밖에는
29997	맘을
29998	맛도
29999	망하지

30000 rows × 1 columns

Enjoy your stylish business and campus life with BIZCAM

4) 허깅페이스 토크나이저 (Huggingface Tokenizer)

4.1) BERT의 워드피스 토크나이저(BertWordPieceTokenizer)

4. 새로운 단어 입력, 인코딩

```
encoded = tokenizer.encode('집현전 발표준비 얼른 해야지')
print(f'토큰화 결과 : {encoded.tokens}')
print(f'정수 인코딩 :{encoded.ids}')    # 딥러닝 모델에 실질적으로 입력되는 정수 인코딩 형태의 결과물을 출력
print(f'디코딩 : {tokenizer.decode(encoded.ids)}')  # 정수 시퀀스를 문자열로 다시 복원하여 출력
```

```
토큰화 결과 : ['집', '##현', '##전', '발', '##표', '##준비', '얼른', '해야지']
정수 인코딩 :[2522, 3741, 3302, 1581, 3585, 21833, 15688, 11786]
디코딩 : 집현전 발표준비 얼른 해야지
```


Enjoy your stylish business and campus life with BIZCAM

4) 허깅페이스 토크나이저 (Huggingface Tokenizer)

4.2) 기타 토크나이저

- BertWordPieceTokenizer : BERT에서 사용된 워드피스 토크나이저(WordPiece Tokenizer)
- CharBPETokenizer : 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('자연어 처리는 정말 재미있습니다.')
print(encoded.tokens)
```

```
['_자연', '어', '_처', '리는', '_정말', '_재미있', '습니다.']
```


Enjoy your stylish business and campus life with BIZCAM

감사합니다!