14. Subword Tokenizer

Subword Tokenizer

- 하나의 단어를 subword로 분리하여 희귀단어, 신조어와 같은 OOV(out of vocabulary) 문제를 완화시킴
- 한글의 경우 조사와 동사 변형과 같은 현상으로 인해 어절 단위로 분리할 경우 서로 다른 단어들이 많아지는 문제를 완화시킬 수 있음
- 형태소 분석을 하지 않고 코퍼스 데이터에서 발생빈도수를 이용하여 단어를 분리하는 방식이 있음
- 기계번역에서는 단어숫자를 적절한 범위에서 통제하는 것이 중요함

한글 데이터 통계

구분	단어의 수	문장수	고유단어 수
위키백과	43.4M	3.3M	299,528
온라인 뉴스	47.1M	3.2M	282,955
세종 말뭉치	31.4M	2.2M	231,332

S. Park *et al*., Subword-level word vector representations for Korean, *Proc. Annual Meeting of Associations for Computational Linguistics*, 2018.

Byte Pair Encoding

- R. Senrich et al., Neural machine translation of rare words with subword units, *Proc. ACL*, 2016.
- 말뭉치에서 많이 등장한 문자열을 병합해 문자열을 압축함
- 영어 사례
  - 데이터: aaabdaaabac
  - 문자열에서 많이 나타난 aa를 Z로 치환: ZabdZabac
  - 여기서 ab를 다시 Y로 치환: ZYdZYac
- **토크나이즈 메커니즘:** 원하는 어휘 집합 크기가 될 때까지 반복적으로 고빈도 문자열들을 병합해 어휘 집합에 추가
- 학습이 끝나면 문장 내 각 어절에 subword가 포함되어 있으면 해당 subword를 어절에서 분리. 어휘 집합에 없으면 미등록 단어로 취급

한글 BPE 사례

- 학습 결과 고빈도 subword가 학교, 밥, 먹었 등이라고 가정 (다음에서 _는 어절의 첫번째 서브워드임을 표시
  - 새로운 문장 분석: 학교에서 밥을 먹었다 → _학교, 에서, _밥, 을, _먹었, 다
- BPE를 수행하면 한글 형태소 분석을 하지 않더라도 주요 단어들을 추출할 수 있음
- Detokenize를 수행시키면 원래 문장으로 바꿀 수 있음

SentencePiece

- T. Kudo and J. Richardson, SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, *Proc. Conf. on Empirical Methods in Natural Language Processing*, 2018.
- BPE를 포함한 subword tokenizing 프로그램을 지원
- 한글과의 번역기에서는 문장을 학습시킬 때 형태소 분석 또는 sentencepiece를 수행시키는 것이 필요함
- 이 프로그램을 사용하려면 다음과 같이 설치

```
> pip install sentencepiece
```

네이버 영화 리뷰 토큰화

- 일반인들이 작성한 영화평들을 읽고 토큰화하는 과정을 수행

```
import pandas as pd
import sentencepiece as spm
import urllib.request
import csv

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') # Null 값이 존재하는 행 제거

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

sentencepiece 실행

- sentencepiece 실행

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

- 실행이 완료되면 naver.model과 naver.vocab 파일들이 생성
- vocab 파일에는 5000개의 서브워드가 집합에 있음

sentencepiece 결과 확인

- naver.model 파일을 이용하여 분석을 실행할 수 있음

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

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]
```

과제 #4 Due: 11/18

제출 방법: 프로그램과 결과를 e-class에 제출

1. 7쪽에서 생성된 naver_review.txt 파일을 읽어서 3쪽에 있는 데이터들을 계산. 단어의 수는 공백으로 분리된 어절의 수를 의미
2. 단어수를 10,000개와 20,000개로 늘려 8쪽의 sentencepiece를 실행하고 80001~80010번의 평에 대한 결과를 얻음
3. naver_review.txt 파일 내용에 대해 Okt 형태소 분석기를 실행시킴. 결과에서 나타난 고유단어수를 계산. 80001~80010번의 평에 대한 결과에 대해 Okt를 실행하고 위 2번의 결과와 비교함.