

# 하위 단어 토큰화

## SUBWORD-BASED TOKENIZATION

p.248-263

---

202284046 김나연

참고자료

<https://wikidocs.net/161976>

<https://youtu.be/zHvTiHr506c?si=DeZe-fQXiN60ArhE>

# 하위 단어 토큰화

---

하나의 단어가 빈번하게 사용되는 하위 단어 조합으로 나누어 토큰화 하는 방법

단어의 길이를 줄이고 유사한 단어 간의 의미를 유지할 수 있도록 도와줌

➡처리 속도 증가,OOV문제, 신조어,은어,고유어 등으로 인한 문제 완화

바이트 페어 인코딩,워드피스,유니그램 모델 등 알고리즘을 사용

ex\_01) transformer -> trans,former

ex\_02) 인공신경망 -> 인공, 신경망

# 바이트 페어 인코딩(BFE)

---

텍스트 데이터에서 가장 빈번하게 등장하는 글자 쌍의 조합을 찾아 부호화하는 알고리즘

자주 등장하는 단어는 하나의 토큰으로 토큰화, 덜 등장하는 단어는 여러 토큰의 조합으로 표현

텍스트 데이터(말뭉치)에서 자주 등장하는 글자 쌍을 찾아 어휘 사전을 구축

ex) abracadabra

1. 'ab'의 빈도수 높음 -> 'ab'를 'A'로 치환

∴ AracadAra

2. 'ra'의 빈도수 높음 -> 'ra'를 'B'로 치환

∴ ABcadAB

3. 'AB'의 빈도수 높음 -> 'AB'를 'C'로 치환

∴ CcadC

- ∴ 1.입력 데이터에서 가장 많이 등장한 글자의 빈도수 측정
- 2.가장 빈도수가 높은 글자 쌍을 탐색

# 바이트 페어 인코딩(BFE)

빈도 사전: (“low”,5), (“lower”,2), (“newest”,6), (“widest”,3)

어휘 사전:[“low”,“lower”,“newest”,“widest”]

## 1.BFE를 적용하기 위해 모든 단어를 글자 단위로 나눔

빈도 사전:(“l”,“o”,“w”,5),(“l”,“o”,“w”,“e”,“r”,2),(“n”,“e”,“w”,“e”,“s”,“t”,6),(“w”,“i”,“d”,“e”,“s”,“t”,3)

어휘 사전: [‘d’,‘e’,‘i’,‘l’, ‘n’, ‘o’, ‘r’, ‘s’, ‘t’,‘w’]

## 2.빈도 사전을 기준 총 9번 등장하는 ‘e’, ‘s’를 병합해 ‘es’를 어휘 사전에 추가

빈도 사전:(“l”,“o”,“w”,5),(“l”,“o”,“w”,“e”,“r”,2),(“n”,“e”,“w”,“es”,“t”,6),(“w”,“i”,“d”,“es”,“t”,3)

어휘 사전: [‘d’,‘e’,‘i’,‘l’, ‘n’, ‘o’, ‘r’, ‘s’, ‘t’,‘w’,‘es’]

## 3.빈도 사전을 기준 총 9번 등장하는 ‘es’, ‘t’를 병합해 ‘est’를 어휘 사전에 추가

빈도 사전:(“l”,“o”,“w”,5),(“l”,“o”,“w”,“e”,“r”,2),(“n”,“e”,“w”,“est”,6),(“w”,“i”,“d”,“est”,3)

어휘 사전: [‘d’,‘e’,‘i’,‘l’, ‘n’, ‘o’, ‘r’, ‘s’, ‘t’,‘w’,‘es’,‘est’]

## 4.위와 같은 과정 반복

∴ 빈도 사전:(“l”,“o”,“w”,5),(“l”,“o”,“w”,“e”,“r”,2),(“n”,“e”,“w”,“est”,6),(“w”,“i”,“d”,“est”,3)

어휘 사전: [‘d’,‘e’,‘i’,‘l’, ‘n’, ‘o’, ‘r’, ‘s’, ‘t’,‘w’,’es’, 'est', lo', low', 'ne', new, 'newest', wi, wid', widest]

기존 말뭉치에 등장하지 않았던 단어가 입력되도 어휘 사전을 참고해 토큰화 가능

# 센텐스피스,코포라

---

## 센텐스피스 라이브러리

- BFE와 유사한 알고리즘 사용하여 입력 데이터를 토큰화라고 단어 사전 생성
- 워드피스,유니코드 기반의 다양한 알고리즘 지원, 세밀한 토큰나이징 기능 제공

## 코포라 라이브러리

- 국립국어원이나 AI Hub에서 제공하는 말뭉치 데이터를 쉽게 사용하도록 제공하는 오픈소스 라이브러리

```
pip install sentencepiece Korpora
```

# 토크나이저 모델 학습

학습 데이터셋 생성(예제 5.11) 토크나이저 모델 학습 (예제 5.12)

[ ]:

```
from Korpora import Korpora
```

```
corpus = Korpora.load("korean_petitions")
```

```
petitions = corpus.get_all_texts()
```

```
with open("../datasets/corpus.txt", "w", encoding="utf-8") as f:
```

```
    for petition in petitions:
```

```
        f.write(petition + "###n")
```

corpus의 get\_all\_texts() 메서드로 본문 데이터셋을 한 번에 불러옴  
하나의 텍스트 파일로 저장

[ ]:

```
from sentencepiece import SentencePieceTrainer
```

```
SentencePieceTrainer.Train(
```

```
    "--input=../datasets/corpus.txt##
```

```
    --model_prefix=../models/petition_bpe##
```

```
    --vocab_size=8000 model_type=bpe"
```

```
)
```

Train 메서드로 토크나이저 모델을 학습

센텐스피스는 문자열 입력을 통해 인자들을 전달받음

∴ 모델 학습이 완료:petition\_bpe.model,petition\_bpe.vocab파일 생성

model 파일:토크나이저 저장된 파일, vocab:어휘 사전이 저장된 파일

# 토큰나이저 모델 학습

바이트 페어 인코딩 토큰화(예제 5.13)

In [ ]:

```
from sentencepiece import SentencePieceProcessor
```

```
tokenizer = SentencePieceProcessor()           SentencePieceProcessor 클래스를 통해 학습된 모델을 불러옴  
tokenizer.load("../models/petition_bpe.model") tokenizer.load()를 사용하여 petition_bpe.model 모델을 불러옴
```

```
sentence = "안녕하세요, 토큰나이저가 잘 학습되었군요!"  
sentences = ["이렇게 입력값을 리스트로 받아서", "쉽게 토큰나이저를 사용할 수 있습니다"]
```

```
tokenized_sentence = tokenizer.encode_as_pieces(sentence)   encode_as_pieces 메서드를 사용해서 문장을 토큰화  
tokenized_sentences = tokenizer.encode_as_pieces(sentences)  
print("단일 문장 토큰화 :", tokenized_sentence)  
print("여러 문장 토큰화 :", tokenized_sentences)
```

```
encoded_sentence = tokenizer.encode_as_ids(sentence)        encode_as_ids 메서드를 통해 토큰을 정수로 인코딩해 제공  
encoded_sentences = tokenizer.encode_as_ids(sentences)     이때, 정수 데이터: 어휘 사전의 토큰에 매핑된 ID 값을 의미  
print("단일 문장 정수 인코딩 :", encoded_sentence)  
print("여러 문장 정수 인코딩 :", encoded_sentences)
```

```
decode_ids = tokenizer.decode_ids(encoded_sentences)        decode_ids, decode_pieces 메서드를 통해 문자열 데이터로 변환  
decode_pieces = tokenizer.decode_pieces(encoded_sentences)  
print("정수 인코딩에서 문장 변환 :", decode_ids)  
print("하위 단어 토큰에서 문장 변환 :", decode_pieces)
```

# 토크나이저 모델 학습

어휘 사전 불러오기(예제 5.14)

```
from sentencepiece import SentencePieceProcessor
```

```
tokenizer = SentencePieceProcessor()  
tokenizer.load("../models/petition_bpe.model")
```

```
vocab = {idx: tokenizer.id_to_piece(idx) for idx in range(tokenizer.get_piece_size())}  
print(list(vocab.items())[:5])  
print("vocab size :", len(vocab))
```

get\_piece\_size 메서드: 센텐스피스 모델에서 생성된 하위 단어의 개수를 반환  
id\_to\_piece 메서드: 정숫값을 하위 단어로 변환하는 메서드

∴ 하위 단어의 개수만큼 반복해서 하위 단어 딕셔너리를 구성



# 워드피스

새로운 하위 단어를 생성할 때 이전 하위 단어와 함께 나타날 확률을 계산해 가장 높은 확률을 가진 하위 단어 선택

$$score = \frac{f(x, y)}{f(x) * f(y)}$$

x와 y를 병합하는 것이 적절한지를  
판단하기 위한 점수

f:빈도를 나타내는 함수  
x,y는 병합하려는 하위 단어  
f(x,y): x와 y가 조합된 글자 쌍의 빈도=xy 글자 쌍의 빈도

# 워드피스

---

빈도사전: (“l”, “o”, “w”, 5), (“l”, “o”, “w”, “e”, “r”, 2), (“n”, “e”, “w”, “e”, “s”, “t”, 6), (“w”, “i”, “d”, “e”, “s”, “t”, 3)

어휘 사전: [‘d’, ‘e’, ‘i’, ‘l’, ‘n’, ‘o’, ‘r’, ‘s’, ‘t’, ‘w’]

가장 빈번하게 등장한 쌍: ‘e’, ‘s’

‘e’, ‘s’: 9번

‘e’: 17번

‘s’: 9번

∴  $\text{score} = 9 / (17 * 9) \doteq 0.06$

➡ 이런 과정을 반복해 연속된 글자 쌍이 더 이상 나타나지 않거나 정해진 어휘 사전 크기에 도달할 때까지 학습한다

# 토크나이저스

---

## 정규화와 사전 토큰화를 제공

정규화: 모호한 경우를 방지하기 위해 일부 문자 대체 제거하는 등의 작업을 수행

사전 토큰화: 문장을 토큰화하기 전에 단어와 같은 작은 단위로 나누는 기능을 제공

공백 혹은 구두점을 기준으로 입력 문장을 나눠 데이터를 효율적으로 처리하고 모델의 성능 향상

```
from tokenizers import Tokenizer
from tokenizers.models import WordPiece
from tokenizers.normalizers import Sequence, NFD, Lowercase
from tokenizers.pre_tokenizers import Whitespace
```

```
tokenizer = Tokenizer(WordPiece())
```

```
tokenizer.normalizer = Sequence([NFD(), Lowercase()])
```

```
tokenizer.pre_tokenizer = Whitespace()
```

```
tokenizer.train(["../datasets/corpus.txt"])
```

```
tokenizer.save("../models/petition_wordpiece.json")
```

토크나이저 라이브러리의 토크나이저로 워드피스 모델을 불러옴

정규화 방식 설정: `normalizers` 모듈에 포함된 클래스를 불러와 시퀀스 형식으로 인스턴스 전달

사전 토큰화 방식 설정: `pre_tokenizer` 모듈에 포함된 클래스를 불러와 적용

훈련(`train`) 메서드를 통해 데이터셋 경로를 전달

학습 완료되면 저장(`save`) 메서드로 학습 결과를 JSON 형태로 저장

# 워드피스 토큰화

In [ ]:

```
from tokenizers import Tokenizer
from tokenizers.decoders import WordPiece as WordPieceDecoder
```

```
tokenizer = Tokenizer.from_file("../models/petition_wordpiece.json")
tokenizer.decoder = WordPieceDecoder()
```

json 파일을 불러와 Tokenizer 객체 생성  
Tokenizer의 디코더를 워드피스 디코더로 설정

```
sentence = "안녕하세요, 토큰라이저가 잘 학습되었군요!"
sentences = ["이렇게 입력값을 리스트로 받아서", "쉽게 토큰라이저를 사용할 수 있습니다"]
```

```
encoded_sentence = tokenizer.encode(sentence)
encoded_sentences = tokenizer.encode_batch(sentences)
```

인코딩 메서드와 인코딩 배치 메서드를 토큰라이저를 통해  
문자를 토큰화, 각 토큰의 색인 번호를 반환

```
print("인코더 형식 :", type(encoded_sentence))
```

```
print("단일 문장 토큰화 :", encoded_sentence.tokens)
print("여러 문장 토큰화 :", [enc.tokens for enc in encoded_sentences])
```

```
print("단일 문장 정수 인코딩 :", encoded_sentence.ids)
print("여러 문장 정수 인코딩 :", [enc.ids for enc in encoded_sentences])
```

```
print("정수 인코딩에서 문장 변환 :", tokenizer.decode(encoded_sentence.ids))
```