**[sentencepiece]**

**토큰화 (tokenization)**

<https://wikidocs.net/21698>

토큰 기준이 word라면, word tokenization

Corpus = 말뭉치 = 언어의 표본

구두점

NLTK - 영어 코퍼스를 토큰화하기 위한 도구들을 제공

KoNLPY - .

\*토큰화에서 고려해야할 사항

토큰화 작업을 단순하게 코퍼스에서 구두점을 제외하고 공백 기준으로 잘라내는 작업이라고 간주할 수는 없다.

보다 섬세한 알고리즘이 필요, 그 이유 3가지 정리

1. 구두점, 특수문자 단순 제외 위험

코퍼스에 대한 정제 작업을 진행하다 보면, 구두점조차도 하나의 토큰으로 분류하기도 합니다. 가장 기본적인 예를 들어보자면, 마침표(.)와 같은 경우는 문장의 경계를 알 수 있는데 도움이 되므로 단어를 뽑아낼 때, 마침표(.)를 제외하지 않을 수 있습니다.

또 다른 예를 들어보면, 단어 자체에서 구두점을 갖고 있는 경우도 있는데, m.p.h나 Ph.D나 AT&T 같은 경우가 있습니다. 또 특수 문자의 달러($)나 슬래시(/)로 예를 들어보면, $45.55와 같은 가격을 의미 하기도 하고, 01/02/06은 날짜를 의미하기도 합니다. 보통 이런 경우 45.55를 하나로 취급해야

숫자 사이에 컴마(,) - 수치를 표현할 때는 123,456,789

1. 줄임말과 단어 내에 띄어쓰기가 있는 경우

아포스트로피(') - 줄임말로 쓰일 때 생기는 형태

하나의 단어이지만 중간에 띄어쓰기 - New York 등 띄어쓰기가 있는 경우에도 하나의 토큰으로 봐야하는 경우

* 문장 토큰화(sentence tokenization)

코퍼스 내에서 문장 단위로 구분하는 작업

문장 분류(sentence segmentation)라고도

직관적으로 생각해봤을 때는 ?나 마침표(.)나 ! 기준으로 문장을 잘라내면 되지 않을까라고 생각할 수 있지만, 꼭 그렇지만은 않습니다. !나 ?는 문장의 구분을 위한 꽤 명확한 구분자(boundary) 역할을 하지만 마침표는 꼭 그렇지 않기 때문입니다. 다시 말해, 마침표는 문장의 끝이 아니더라도 등장할 수 있습니다.

**192.168.56.31 gmail.com Ph.D. 등**

그렇기 때문에 사용하는 코퍼스가 어떤 국적의 언어인지, 또는 해당 코퍼스 내에서 특수문자들이 어떻게 사용되고 있는지에 따라서 직접 규칙들을 정의해볼 수 있겠습니다. 물론, 100% 정확도를 얻는 일은 쉬운 일이 아닙니다. 갖고있는 코퍼스 데이터에 오타나, 문장의 구성이 엉망이라면 정해놓은 규칙이 소용이 없을 수 있기 때문입니다.

* **이진 분류기 (binary classifier)**

문장 토큰화에서의 예외 사항을 발생시키는 마침표의 처리를 위해서 입력에 따라 두 개의 클래스로 분류하는 이진 분류기(binary classifier)를 사용하기도

1. 마침표(.)가 단어의 일부분일 경우. 즉, 마침표가 약어(abbreivation)로 쓰이는 경우  
2. 마침표(.)가 정말로 문장의 구분자(boundary)일 경우

* 한국어 토큰화의 어려움

영어: New York과 같은 합성어나 he's 와 같이 줄임말에 대한 예외처리만 한다면, 띄어쓰기(whitespace)를 기준으로 하는 띄어쓰기 토큰화를 수행해도 단어 토큰화가 잘 작동

대부분의 경우가 (띄어쓰기 토큰화==단어 토큰화)

한국어: 교착어 (어절 토큰화!=단어 토큰화)

**서브워드 토크나이저 (Subword Tokenizer)**

<https://wikidocs.net/86657>

서브워드 분리(Subword segmenation) 작업은 하나의 단어는 더 작은 단위의 의미있는 여러 서브워드들(Ex) birthplace = birth + place)의 조합으로 구성된 경우가 많기 때문에, 하나의 단어를 여러 서브워드로 분리해서 단어를 인코딩 및 임베딩하겠다는 의도를 가진 전처리 작업

실제로 언어의 특성에 따라 영어권 언어나 한국어는 서브워드 분리를 시도했을 때 어느정도 의미있는 단위로 나누는 것이 가능  
OOV 문제를 완화하는 대표적인 서브워드 분리 알고리즘인 BPE(Byte Pair Encoding) 알고리즘을 소개하고, 다음 챕터에서 이 알고리즘을 실무에서 사용할 수 있도록 구현한 센텐스피스(Sentencepiece)를 소개

**BPE(Byte Pair Encoding**)

1994년에 제안된 데이터 압축 알고리즘

 자연어 처리의 서브워드 분리 알고리즘으로 응용

* 기존의 BPE의 작동 방법

Most often byte pair -> replaced by a byte

연속적으로 가장 많이 등장한 글자의 쌍을 찾아서 하나의 글자로 병합하는 방식으로 압축을 수행

aaabdaaabac -> ZYdZYac -> XdXac

* **WPM (WordPiece Model)**

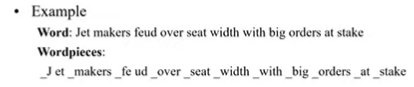
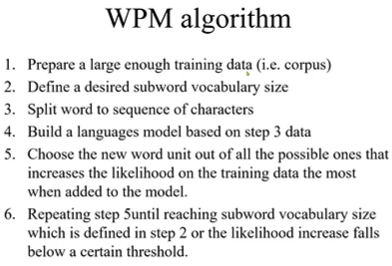
BPE의 변형 알고리즘

WPM은 BPE가 빈도수에 기반하여 가장 많이 등장한 쌍을 병합하는 것과는 달리, 병합되었을 때 코퍼스의 우도(Likelihood)를 가장 높이는 쌍을 병합

* Problem : BOW(Bag-of-words) model

High dimension (embedding)

Ignore low freq. words -> OOV problem

* 딥러닝 성능 저하 문제발생
* solution : word representation by ‘**subword units**’(word단위 no, subword단위로)
* 
* 
* 

파란줄 : 출연빈도가 높아서 (blank는 \_)

* **nlp에서의 BPE**

논문 : <https://arxiv.org/pdf/1508.07909.pdf>

자연어 처리에서의 BPE는 서브워드 분리(subword segmentation) 알고리즘

= 기존에 있던 단어를 분리한다는 의미

글자(charcter) 단위에서 점차적으로 단어 집합(vocabulary)을 만들어 내는 Bottom up 방식의 접근

 훈련 데이터에 있는 단어들을 모든 글자(chracters) 또는 유니코드(unicode) 단위로 단어 집합(vocabulary)를 만들고, 가장 많이 등장하는 유니그램을 하나의 유니그램으로 통합

**센텐스피스 (SentencePiece toknizer)**

: unigram language model

논문 : <https://arxiv.org/pdf/1808.06226.pdf>

센텐스피스 깃허브 : <https://github.com/google/sentencepiece>

BPE 알고리즘과 Unigram Language Model Tokenizer를 구현한 센텐스피스

내부 단어 분리를 위한 유용한 패키지

Uni

* **N-gram language model**

https://wikidocs.net/21692

count based 통계적 접근 model, SLM 일종

다만, 이전에 등장한 모든 단어 고려x -> 일부 단어만 고려하는 접근법 -> N개의 단어만 고려하자

**N-gram ?**

N개의 연속적인 단어의 나열, corpus에서 n개의 단어 뭉치를 단위로 끊어 이를 하나의 token으로 간주 (예측 target 앞, n-1개 고려, 모든 조합을 고려한다며.. 그건 또 무슨 소리)

**uni**grams : an, adorable, little, boy   
**bi**grams : an adorable, adorable little, little boy   
**tri**grams : an adorable little, adorable little boy  
**4**-grams : an adorable little boy

* Assumption

A sentence is “a sequence of subwords”



X라는 word는 M개의 subword(x)들로 구성되어 있음

M개의 subword의 후보들 중 어떠한 조합을 선택? 이 관건 -> by unigram model 선택!