공백 기반 토큰화

한 문장에서 단어의 수는 어떻게 정의할 수 있을까요? `"그녀는 나와 밥을 먹는다"` 라는 문장이 주어지면 공백 기준으로 나누어 `1: 그녀는 2: 나와 3: 밥을 4: 먹는다` 4개 단어로 이루어졌다고 단정 지을 수 있을까요? 어쩌면 `1: 그녀 2: 는 3: 나 4: 와 5: 밥 6: 을 7: 먹는다` 처럼 잘게 잘게 쪼개어 7개 단어로 이루어졌다고 할 수도 있겠죠? 그것은 우리가 정의할 **토큰화 기법이 결정할 부분**이랍니다!

문장을 어떤 기준으로 쪼개었을 때, 쪼개진 각 단어들을 **토큰(Token)** 이라고 부릅니다. 그리고 그 쪼개진 기준이 **토큰화(Tokenization) 기법**에 의해 정해지죠. 이번 스텝에서는 토큰화의 여러 가지 기법에 대해 배워보도록 하겠습니다.

자연어의 노이즈를 제거하는 방법 중 하나로 우리는 `Hi,` 를 `Hi` 와 `,` 로 나누기 위해 문장부호 양옆에 공백을 추가해 주었습니다. 그것은 이 **공백 기반 토큰화**를 사용하기 위해서였죠! 당시의 예제 코드를 다시 가져와 공백을 기반으로 토큰화를 진행해 보겠습니다.

```
corpus = \
"""
in the days that followed i learned to spell in this uncomprehending way a great
but my teacher had been with me several weeks before i understood that everythin
one day ,  we walked down the path to the well house ,  attracted by the fragran
some one was drawing water and my teacher placed my hand under the spout .
as the cool stream gushed over one hand she spelled into the other the word wate
i stood still ,  my whole attention fixed upon the motions of her fingers .
suddenly i felt a misty consciousness as of something forgotten a thrill of retu
i knew then that  w a t e r  meant the wonderful cool something that was flowing
that living word awakened my soul ,  gave it light ,  hope ,  joy ,  set it free
there were barriers still ,  it is true ,  but barriers that could in time be sw
"""
# HINT : split()을 사용하여 공백토큰화를 수행하세요.
tokens = corpus.split()

print("문장이 포함하는 Tokens:", tokens)
```

```
문장이 포함하는 Tokens: ['in', 'the', 'days', 'that', 'followed', 'i', 'learned',
```

형태소 기반 토큰화

하지만 우리에겐 영어 문장이 아닌 한국어 문장을 처리할 일이 더 많을 것이고, 한국어 문장은 **공백 기준**으로 토큰화를 했다간 **엉망진창의 단어들이 등장하는 것**을 알 수 있습니다. 문장부호처럼 **"은 / 는 / 이 / 가"** 양옆에 공백을 붙이자구요? 글쎄요... 가로 시작하는 단어만 해도 가면, 가위, 가족, 가수... 의도치 않은 변형이 너무나도 많이 일어날 것 같네요!

이를 어떻게 해결할 수 있을까요? 정답은 **형태소**에 있습니다. 어릴 적 국어 시간에 배운 기억이 새록새록 나시나요? 상기시켜드리면 형태소의 정의는 아래와 같습니다.

(명사) 뜻을 가진 가장 작은 말의 단위.

예를 들어, 오늘도 공부만 한다 라는 문장이 있다면, 오늘, 도, 공부, 만, 한다 로 쪼개지는 것이 바로 형태소죠. 한국어는 이를 활용해 토큰화를 할 수 있습니다!

한국어 형태소 분석기는 대표적으로 아래 두 가지가 사용됩니다.

- KoNLPy 파이썬 한국어 패키지
- kakao/khaiii

KoNLPy는 내부적으로 5가지의 형태소 분석 Class를 포함하고 있습니다. Khaiii까지 총 6개나 되는 형태소 분석기들은 **특수한 문장(띄어쓰기 X / 오탈자) 처리 성능, 속도** 측면에서 **차이**를 보입니다. 천하무적인 것은 (아직은) 없으니, 각 분석기를 직접 테스트해보고 적합한 것을 선택해 사용하면 됩니다.

Colab에서 mecab설치하기

```
1 !git clone https://github.com/SOMJANG/Mecab-ko-for-Google-Colab.git
```

```
Cloning into 'Mecab-ko-for-Google-Colab'...
remote: Enumerating objects: 91, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 91 (delta 0), reused 0 (delta 0), pack-reused 90
Unpacking objects: 100% (91/91), done.
```

```
1 !pip install konlpy
```

```
Collecting konlpy
  Downloading konlpy-0.5.2-py2.py3-none-any.whl (19.4 MB)
     |████████████████████████████████| 19.4 MB 1.3 MB/s
Requirement already satisfied: lxml>=4.1.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: numpy>=1.6 in /usr/local/lib/python3.7/dist-pac
Collecting colorama
  Downloading colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Collecting JPype1>=0.7.0
  Downloading JPype1-1.3.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.w
     |████████████████████████████████| 448 kB 51.8 MB/s
Collecting beautifulsoup4==4.6.0
  Downloading beautifulsoup4-4.6.0-py3-none-any.whl (86 kB)
     |████████████████████████████████| 86 kB 5.2 MB/s
Requirement already satisfied: tweepy>=3.7.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/d
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/pyth
Requirement already satisfied: requests[socks]>=2.11.1 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/d
```

```
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python
Installing collected packages: JPype1, colorama, beautifulsoup4, konlpy
  Attempting uninstall: beautifulsoup4
    Found existing installation: beautifulsoup4 4.6.3
    Uninstalling beautifulsoup4-4.6.3:
      Successfully uninstalled beautifulsoup4-4.6.3
Successfully installed JPype1-1.3.0 beautifulsoup4-4.6.0 colorama-0.4.4 konlpy
```

```
1 cd Mecab-ko-for-Google-Colab/
```

```
/content/Mecab-ko-for-Google-Colab
```

```
1 ! bash install_mecab-ko_on_colab190912.sh
```

```
Installing konlpy.....
Requirement already satisfied: konlpy in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: JPype1>=0.7.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: lxml>=4.1.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: numpy>=1.6 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: colorama in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: beautifulsoup4==4.6.0 in /usr/local/lib/python3
Requirement already satisfied: tweepy>=3.7.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/d
Requirement already satisfied: requests[socks]>=2.11.1 in /usr/local/lib/pytho
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/pyth
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/d
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python
Done
Installing mecab-0.996-ko-0.9.2.tar.gz.....
Downloading mecab-0.996-ko-0.9.2.tar.gz.......
from https://bitbucket.org/eunjeon/mecab-ko/downloads/mecab-0.996-ko-0.9.2.tar
--2021-11-26 07:19:21--  https://bitbucket.org/eunjeon/mecab-ko/downloads/meca
Resolving bitbucket.org (bitbucket.org)... 104.192.141.1, 2406:da00:ff00::3403
Connecting to bitbucket.org (bitbucket.org)|104.192.141.1|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://bbuseruploads.s3.amazonaws.com/eunjeon/mecab-ko/downloads/me
--2021-11-26 07:19:22--  https://bbuseruploads.s3.amazonaws.com/eunjeon/mecab-
Resolving bbuseruploads.s3.amazonaws.com (bbuseruploads.s3.amazonaws.com)... 5
Connecting to bbuseruploads.s3.amazonaws.com (bbuseruploads.s3.amazonaws.com)|
HTTP request sent, awaiting response... 200 OK
Length: 1414979 (1.3M) [application/x-tar]
Saving to: ‘mecab-0.996-ko-0.9.2.tar.gz’

mecab-0.996-ko-0.9. 100%[===================>]   1.35M  2.78MB/s    in 0.5s

2021-11-26 07:19:23 (2.78 MB/s) - ‘mecab-0.996-ko-0.9.2.tar.gz’ saved [1414979

Done
Unpacking mecab-0.996-ko-0.9.2.tar.gz.......
Done
```

```
Change Directory to mecab-0.996-ko-0.9.2.......
installing mecab-0.996-ko-0.9.2.tar.gz........
configure
make
make check
make install
ldconfig
Done
Change Directory to /content
Downloading mecab-ko-dic-2.1.1-20180720.tar.gz.......
from https://bitbucket.org/eunjeon/mecab-ko-dic/downloads/mecab-ko-dic-2.1.1-2
--2021-11-26 07:20:55--  https://bitbucket.org/eunjeon/mecab-ko-dic/downloads/
Resolving bitbucket.org (bitbucket.org)... 104.192.141.1, 2406:da00:ff00::22c5
Connecting to bitbucket.org (bitbucket.org)|104.192.141.1|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://bbuseruploads.s3.amazonaws.com/a4fcd83e-34f1-454e-a6ac-c242c
--2021-11-26 07:20:55--  https://bbuseruploads.s3.amazonaws.com/a4fcd83e-34f1-
```

```
1 from konlpy.tag import Mecab
```

```
1 mecab = Mecab()
```

```
1 # None자리에 문장을 넣어보고 토큰화 결과를 출력해보세요.
2
3 # 예시문장 : 자연어처리가너무재밌어서밥먹는것도가끔까먹어요
4 print(mecab.morphs('자연어처리가너무재밌어서밥먹는것도가끔까먹어요'))
```

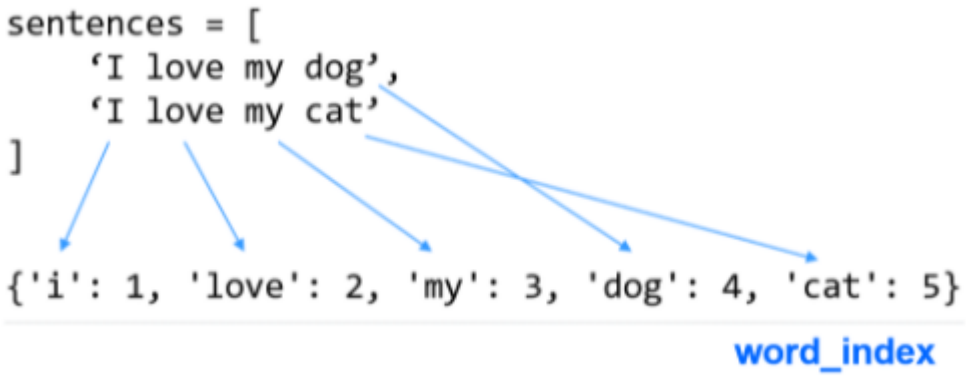
```
['자연어', '처리', '가', '너무', '재밌', '어서', '밥', '먹', '는', '것', '도', '가끔', '
```

```
1 from konlpy.tag import Hannanum,Kkma,Komoran,Mecab,Okt
2 tokenizer_list = [Hannanum(),Kkma(),Komoran(),Mecab(),Okt()]
3
4 kor_text = '코로나바이러스는 2019년 12월 중국 우한에서 처음 발생한 뒤 전 세계로 확산된, 새로운 유형
5
6 for tokenizer in tokenizer_list:
7     print('[{}] \n{}'.format(tokenizer.__class__.__name__, tokenizer.pos(kor_tex
```

```
[Hannanum]
[('코로나바이러스', 'N'), ('는', 'J'), ('2019년', 'N'), ('12월', 'N'), ('중국', 'N'),
[Kkma]
[('코로나', 'NNG'), ('바', 'NNG'), ('이러', 'MAG'), ('슬', 'VV'), ('는', 'ETD'), (
[Komoran]
[('코로나바이러스', 'NNP'), ('는', 'JX'), ('2019', 'SN'), ('년', 'NNB'), ('12월', '1
[Mecab]
[('코로나', 'NNP'), ('바이러스', 'NNG'), ('는', 'JX'), ('2019', 'SN'), ('년', 'NNBC
[Okt]
[('코로나바이러스', 'Noun'), ('는', 'Josa'), ('2019년', 'Number'), ('12월', 'Number
```

인코딩

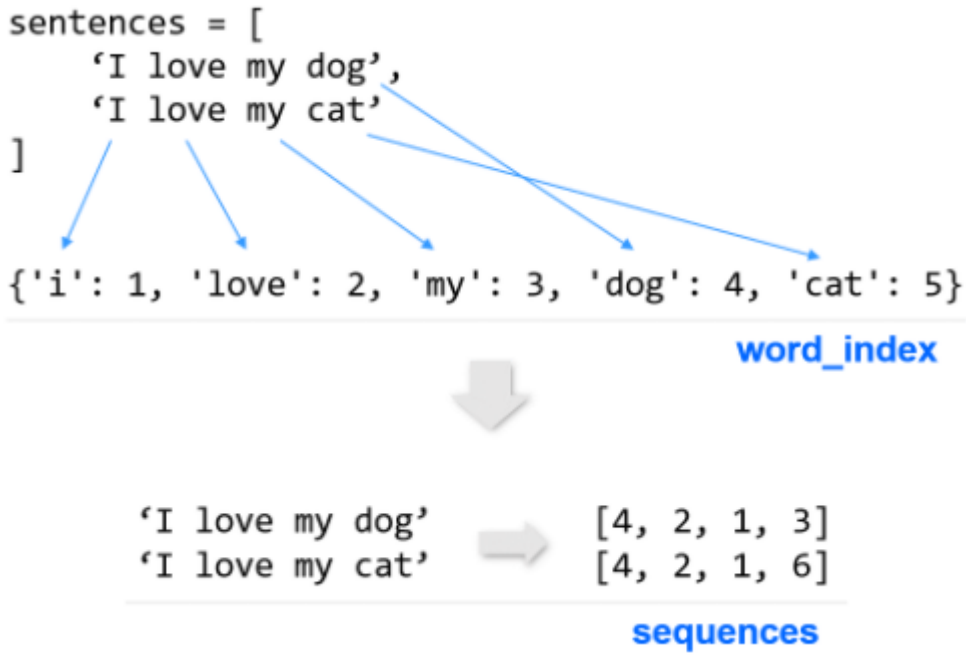
단어 기반 인코딩




```
1 import tensorflow as tf
2 from tensorflow import keras
3 from tensorflow.keras.preprocessing.text import Tokenizer
4
5 sentences = [
6   'I love my dog',
7   'I love my cat',
8 ]
9
10 tokenizer = Tokenizer(num_words = 100)
11 tokenizer.fit_on_texts(sentences) # 문자 데이터를 입력받아서 리스트의 형태로 변환
12 word_index = tokenizer.word_index # 토큰별 단어에 index를 매핑시켜준다.
13 print(word_index)
```

```
{'i': 1, 'love': 2, 'my': 3, 'dog': 4, 'cat': 5}
```

텍스트를 시퀀스로 변환하기

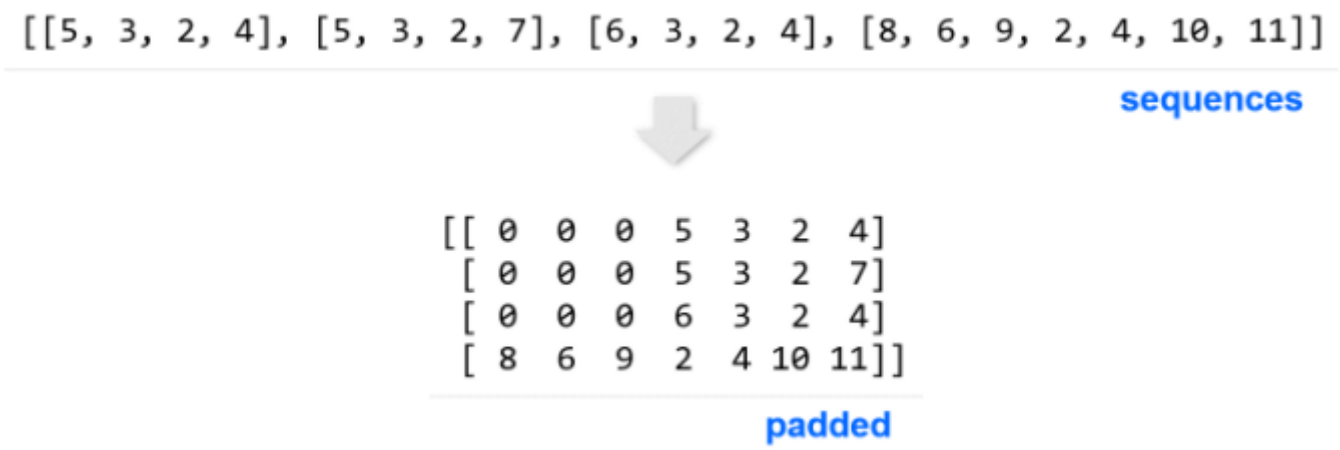



```
1 import tensorflow as tf
2 from tensorflow import keras
```

```
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
  'I love my dog',
  'I love my cat',
  'You love my dog!',
  'Do you think my dog is amazing?'
]

tokenizer = Tokenizer(num_words = 100)
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(sentences) # 텍스트를 시퀀스로 변환

print(word_index)
print(sequences)
```

```
{'my': 1, 'love': 2, 'dog': 3, 'i': 4, 'you': 5, 'cat': 6, 'do': 7, 'think': 8
[[4, 2, 1, 3], [4, 2, 1, 6], [5, 2, 1, 3], [7, 5, 8, 1, 3, 9, 10]]
```

패딩 설정하기




```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

sentences = [
  'I love my dog',
  'I love my cat',
  'You love my dog!',
  'Do you think my dog is amazing?'
]

tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
```

```
sequences = tokenizer.texts_to_sequences(sentences)

padded = pad_sequences(sequences)

print(word_index)
print(sequences)
print(padded)
```

```
{'<OOV>': 1, 'my': 2, 'love': 3, 'dog': 4, 'i': 5, 'you': 6, 'cat': 7, 'do': 8
[[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]
[[ 0  0  0  5  3  2  4]
 [ 0  0  0  5  3  2  7]
 [ 0  0  0  6  3  2  4]
 [ 8  6  9  2  4 10 11]]
```

한글-영어 코퍼스(실제 데이터로)로 토큰화수행하기

```
import os

import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
```

```
path_to_file = '/content/drive/MyDrive/Colab Notebooks/영우4기_자연어/dataset/korea
```

```
with open(path_to_file, "r", encoding='utf-8') as f:
  raw = f.read().splitlines()

print("Data Size", len(raw))
print("Example:")
for sen in raw[0:100][::20]: print(">>", sen)
```

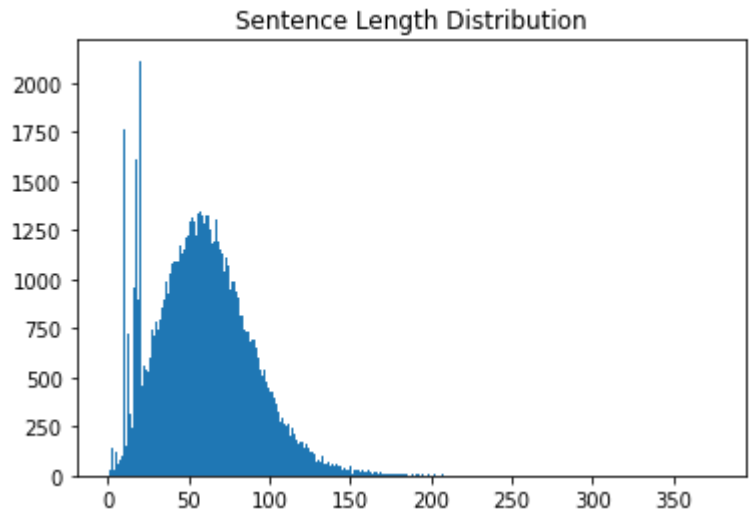
```
Data Size 94123
Example:
>> 개인용 컴퓨터 사용의 상당 부분은 "이것보다 뛰어날 수 있느냐?"
>> 북한의 핵무기 계획을 포기하도록 하려는 압력이 거세지고 있는 가운데, 일본과 북한의 외교관들이 외교
>> "경호 로보트가 침입자나 화재를 탐지하기 위해서 개인적으로, 그리고 전문적으로 사용되고 있습니다."
>> 수자원부 당국은 논란이 되고 있고, 막대한 비용이 드는 이 사업에 대해 내년에 건설을 시작할 계획이다.
>> 또한 근력 운동은 활발하게 걷는 것이나 최소한 20분 동안 뛰는 것과 같은 유산소 활동에서 얻는 운동 호
```

```
min_len = 999
max_len = 0
sum_len = 0

for sen in raw:
    length = len(sen)
    if min_len > length : min_len = length
    if max_len < length : max_len = length
    sum_len += length

print("문장의 최단 길이 :",min_len)
```

```
print("문장의 최장 길이 :", max_len)
print("문장의 평균 길이 :", sum_len//len(raw))
```

```
문장의 최단 길이 : 1
문장의 최장 길이 : 377
문장의 평균 길이 : 60
```

```
sentence_length = np.zeros((max_len), dtype=np.int)
for sen in raw:
    sentence_length[len(sen)-1] += 1

plt.bar(range(max_len), sentence_length, width =1.0)
plt.title("Sentence Length Distribution")
plt.show()
```




```
def check_sentence_with_length(raw, length):
    count = 0

    for sen in raw:
        if len(sen) == length:
            print(sen)
            count += 1
            if count > 100: return

check_sentence_with_length(raw, 1)
```

```
,
```

```
for idx, _sum in enumerate(sentence_length):
    if _sum > 1500:
        print("Outlier Index :", idx+1)
```

```
Outlier Index : 11
Outlier Index : 19
Outlier Index : 21
```

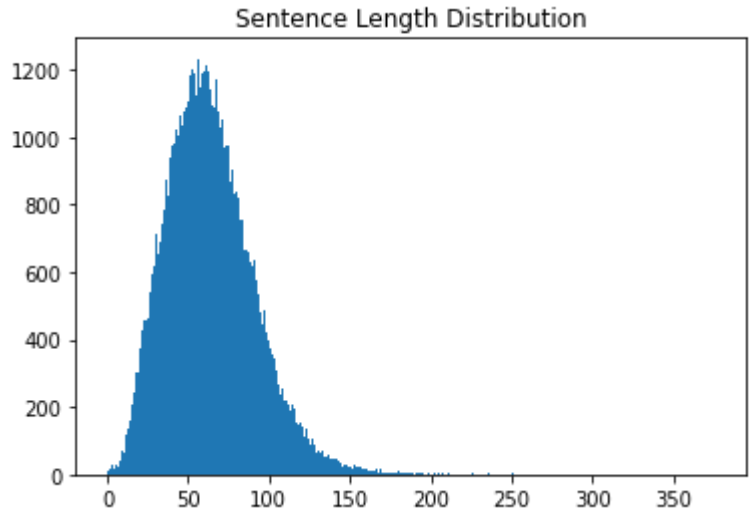
```
check_sentence_with_length(raw, 11)
```

```
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원

경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
케냐 야생동물 고아원
경유 1200원대로…
더 내려야 하는 이유
```

```
min_len = 999
max_len = 0
sum_len = 0

cleaned_corpus = list(set(raw)) # set 사용해서 중복 제거!!
print("Data Size :", len(cleaned_corpus))

for sen in cleaned_corpus:
    length = len(sen)
    if min_len > length : min_len = length
    if max_len < length : max_len = length
    sum_len += length

print("문장의 최단 길이 :",min_len)
print("문장의 최장 길이 :", max_len)
print("문장의 평균 길이 :", sum_len//len(cleaned_corpus))
```

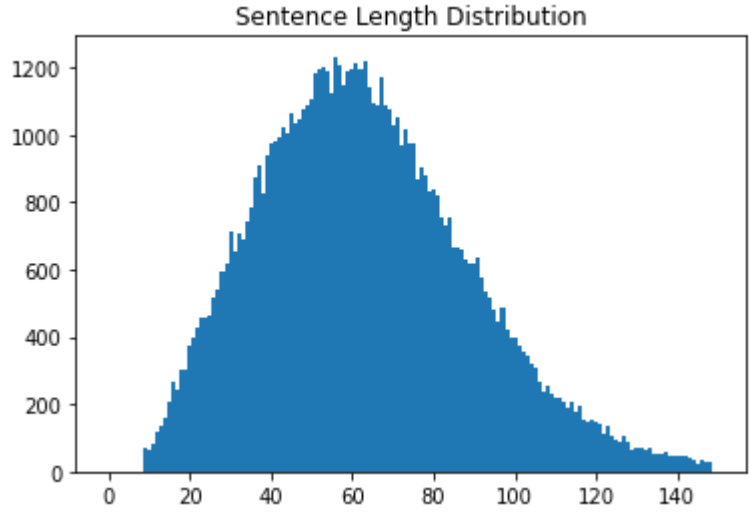
```
Data Size : 77591
문장의 최단 길이 : 1
문장의 최장 길이 : 377
문장의 평균 길이 : 64
```

```
sentence_length = np.zeros((max_len), dtype=np.int)
for sen in cleaned_corpus: # 중복이 제거된 코퍼스
    sentence_length[len(sen)-1] += 1

plt.bar(range(max_len), sentence_length, width =1.0)
plt.title("Sentence Length Distribution")
plt.show()
```




```
max_len = 150
min_len = 10

filtered_corpus = [s for s in cleaned_corpus if (len(s) < max_len) & (len(s) >=

sentence_length = np.zeros((max_len), dtype=np.int)
for sen in filtered_corpus: # 필터가 적용된 코퍼스
    sentence_length[len(sen)-1] += 1

```

```
plt.bar(range(max_len), sentence_length, width =1.0)
plt.title("Sentence Length Distribution")
plt.show()
```



공백 기반 토큰화

```
# Quiz : 정제된 데이터를 공백 기반으로 토큰화하여 list에 저장한 후, tokenize()함수를 사용해 단어 
# 그리고 단어 사전의 크기를 확인하세요.

def tokenize(corpus):

    tokenizer = tf.keras.preprocessing.text.Tokenizer(filters='')
    tokenizer.fit_on_texts(corpus) # 문자 -> 리스트

    tensor = tokenizer.texts_to_sequences(corpus)  # 텍스트 -> 시퀀스
    tensor = tf.keras.preprocessing.sequence.pad_sequences(tensor, padding='post

    return tensor, tokenizer
```

```
# 정제된 데이터를 공백 기반으로 토큰화하여 저장하는 코드를 직접 작성해보세요.
split_corpus = []

for kor in filtered_corpus:
    split_corpus.append(kor.split())
    # 코드작성해주세요.
```

```
split_tensor, split_tokenizer = tokenize(split_corpus)
print("Split Vocab Size :", len(split_tokenizer.index_word))
```

```
Split Vocab Size : 237435
```

```
for idx, word in enumerate(split_tokenizer.word_index):
    print(idx, ":", word)

    if idx > 10: break
```

```
0 : 이
1 : 밝혔다.
2 : 있다.
3 : 말했다.
4 : 수
5 : 있는
6 : 그는
7 : 대한
8 : 위해
9 : 전했다.
10 : 지난
11 : 이번
```

형태소 기반 토큰화

```
# 위에서 사용한 코드를 활용해 Mecab단어 사전을 만들어주세요.
# Hint mecab.morphs() --> 형태소 분석 수행

def mecab_split(sentence): # 형태소 분석역할
    return mecab.morphs(sentence)

mecab_corpus = []

# mecab 단어장 생성
for kor in filtered_corpus:
    mecab_corpus.append(mecab_split(kor)) # 형태소 분석 '텍스트'형태
```

```
mecab_tensor, mecab_tokenizer = tokenize(mecab_corpus) # 텍스트 ---> 숫자
print("Mecab Vocab size :", len(mecab_tokenizer.index_word))
```

```
Mecab Vocab size : 52279
```

1) tokenizer.sequences_to_texts() 함수를 사용하여 Decoding

2) tokenizer.index_word 를 사용하여 Decoding

두 가지 방법으로 mecab_tensor[100] 을 원문으로 되돌려 보세요! (여기서 띄어쓰기는 고려하지 않습니다!)

```
# Case 1 : mecab_tokenizer.sequences_to_texts()

# Case 1
texts = mecab_tokenizer.sequences_to_texts([mecab_tensor[100]]) # 코드 작성

print(texts[0])
```

```
중국 관영 언론 은 2000 가지 음식 명 을 제시 한 170 페이지 분량 의 책 을 베이징 호텔 에 제공 했 다
```

```
# Case 2 : mecab_tokenizer.index_word[]
sentence = ""

for w in mecab_tensor[100]:
```

```
    if w == 0: continue
    sentence += mecab_tokenizer.index_word[w] + " "# 코드 작성

print(sentence)
```

```
중국 관영 언론 은 2000 가지 음식 명 을 제시 한 170 페이지 분량 의 책 을 베이징 호텔 에 제공 했 다
```

사전에 없는 단어의 문제

```
코로나바이러스는 2019년 12월 중국 우한에서 처음 발생한 뒤
전 세계로 확산된, 새로운 유형의 호흡기 감염 질환입니다.

→

<unk>는 2019년 12월 중국 <unk>에서 처음 발생한 뒤
전 세계로 확산된, 새로운 유형의 호흡기 감염 질환입니다.
```

만약 위 문장을 영문으로 번역해야 한다면 어떨까요? 핵심인 단어 `코로나바이러스`와 `우한`을 모른다면 제대로 해낼 수 있을 리가 없습니다. 이를 **OOV(Out-Of-Vocabulary)** 문제라고 합니다. 이처럼 **새로 등장한(본 적 없는) 단어에 대해 약한 모습**을 보일 수밖에 없는 기법들이기에, 이를 해결하고자 하는 시도들이 있었습니다. 그리고 그것이 우리가 다음 스텝에서 배울, ***Wordpiece Model***이죠!

```
1
```