NLP Preprocessing

```
import warnings
warnings.filterwarnings('ignore')
```

I. Tokenization

1) 영어 : NLTK(Natural Language Toolkit)

```
import nltk

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

(1) 문장 토큰화 : sent_tokenize( )

```
from nltk import sent_tokenize

sentences = 'The X-Files is an American science fiction drama television series ₩
created by Chris Carter. ₩
             The original television series aired from September 10, 1993 ₩
to May 19, 2002 on Fox. ₩
             The program spanned nine seasons, with 202 episodes.'

sent_tokenize(sentences)
```

```
['The X-Files is an American science fiction drama television series created by Chris Carter.
 'The original television series aired from September 10, 1993 to May 19, 2002 on Fox.',
 'The program spanned nine seasons, with 202 episodes.']
```

(2) 단어 토큰화 : word_tokenize( )

```
from nltk.tokenize import word_tokenize
```

```
text = 'The truth is out there.'

word_tokenize(text)
word_tokenize(text)
```

```
['The', 'truth', 'is', 'out', 'there', '.']
```

(3) 단어 품사(Part Of Speech) 태깅 : pos_tag( )

```
from nltk.tag import pos_tag
```

```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

```
x = word_tokenize(text)

pos_tag(x)
```

```
[('The', 'DT'),
 ('truth', 'NN'),
 ('is', 'VBZ'),
 ('out', 'RP'),
 ('there', 'RB'),
 ('.', '.')]
```

(4) Stop Words

- Import Package and Download 'stopwords'

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

- 'English' Stop Words

```
print('English stop words :',len(nltk.corpus.stopwords.words('english')))
print(nltk.corpus.stopwords.words('english')[:20])
```

```
English stop words : 179
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you
```

- tokenize_text( ) 정의
    - 여러개의 문장별로 단어 토큰 생성 함수 정의

```
from nltk import word_tokenize, sent_tokenize

def tokenize_text(doc):
    sentences = sent_tokenize(doc)
    word_tokens = [word_tokenize(sentence) for sentence in sentences]
    return word_tokens
```

- 문장별 단어 토큰화 수행

```
word_tokens = tokenize_text(sentences)

print(type(word_tokens),len(word_tokens))
```

```
<class 'list'> 3
```

- 문장별 단어 토큰화 결과 확인

```
print(word_tokens)
```

```
[['The', 'X-Files', 'is', 'an', 'American', 'science', 'fiction', 'drama', 'television', 'ser
```

- Stop Words 제거

```
stopwords = nltk.corpus.stopwords.words('english')
all_tokens = []

for sentence in word_tokens:
    filtered_words = []

    for word in sentence:
        word = word.lower()
        if word not in stopwords:
            filtered_words.append(word)
    all_tokens.append(filtered_words)
```

- Stop Words 처리 결과

```
print(all_tokens)
```

```
[['x-files', 'american', 'science', 'fiction', 'drama', 'television', 'series', 'created', 'c
```

(5) Stemming(어간 추출)

- 변화된 단어의 원형으로 처리

- work
- amuse
- happy
- fancy

```
from nltk.stem import LancasterStemmer

stemmer = LancasterStemmer()

print(stemmer.stem('working'),stemmer.stem('works'),stemmer.stem('worked'))
print(stemmer.stem('amusing'),stemmer.stem('amuses'),stemmer.stem('amused'))
print(stemmer.stem('happier'),stemmer.stem('happiest'))
print(stemmer.stem('fancier'),stemmer.stem('fanciest'))
```

```
work work work
amus amus amus
happy happiest
fant fanciest
```

(6) Lemmatization(표제어 추출)

- 변화된 단어의 원형을 처리
  - Stemming 보다 정확한 처리 가능
  - '품사'를 지정하여 사용

```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
True
```

- 'v' 동사, 'a' 형용사

```
from nltk.stem import WordNetLemmatizer

lemma = WordNetLemmatizer()

print(lemma.lemmatize('amusing', 'v'), lemma.lemmatize('amuses', 'v'), ₩
      lemma.lemmatize('amused', 'v'))
print(lemma.lemmatize('happier', 'a'), lemma.lemmatize('happiest', 'a'))
print(lemma.lemmatize('fancier', 'a'), lemma.lemmatize('fanciest', 'a'))
```

```
amuse amuse amuse
happy happy
fancy fancy
```

2) 한국어 : KoNLPy

(1) KoNLPy 패키지 설치

```
!pip install konlpy
```

```
Collecting konlpy
  Downloading https://files.pythonhosted.org/packages/85/0e/f385566fec837c0b83f216b2da65db999
     |████████████████████████████████| 19.4MB 1.5MB/s
Requirement already satisfied: lxml>=4.1.0 in /usr/local/lib/python3.7/dist-packages (from kc
Requirement already satisfied: tweepy>=3.7.0 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: numpy>=1.6 in /usr/local/lib/python3.7/dist-packages (from kor
Collecting beautifulsoup4==4.6.0
  Downloading https://files.pythonhosted.org/packages/9e/d4/10f46e5cfac773e22707237bfcd51bbff
     |████████████████████████████████| 92kB 6.2MB/s
Collecting JPype1>=0.7.0
  Downloading https://files.pythonhosted.org/packages/cd/a5/9781e2ef4ca92d09912c4794642c1653a
     |████████████████████████████████| 460kB 51.0MB/s
Collecting colorama
  Downloading https://files.pythonhosted.org/packages/44/98/5b86278fbbf250d239ae0ecb724f8572a
Requirement already satisfied: requests[socks]>=2.11.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from tw
Requirement already satisfied: typing-extensions; python_version < "3.8" in /usr/local/lib/py
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from r
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/pyth
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6; extra == "socks" in /usr/local/lib/pyt
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (frc
Installing collected packages: beautifulsoup4, JPype1, colorama, konlpy
  Found existing installation: beautifulsoup4 4.6.3
    Uninstalling beautifulsoup4-4.6.3:
      Successfully uninstalled beautifulsoup4-4.6.3
Successfully installed JPype1-1.2.1 beautifulsoup4-4.6.0 colorama-0.4.4 konlpy-0.5.2
```

(2) Okt 형태소 분석기(Open Korea Text, Twitter)

- 형태소(Morpheme)

```
from konlpy.tag import Okt
```

- 토큰화 : okt.morphs( )

```
okt = Okt()

print(okt.morphs('지난 몇 달간 전 세계 모든 사람은 코로나19로 인해 ₩
전례 없는 고통을 겪으며 다양한 방식으로 심각하게 피해를 겪었습니다.'))
```

```
['지난', '몇', '달', '간', '전', '세계', '모든', '사람', '은', '코로나', '19', '로', '인해',
```

- 품사 태깅 : okt.pos( )

```
print(okt.pos('지난 몇 달간 전 세계 모든 사람은 코로나19로 인해 ₩
전례 없는 고통을 겪으며 다양한 방식으로 심각하게 피해를 겪었습니다.'))
```

```
    [('지난', 'Noun'), ('몇', 'Noun'), ('달', 'Noun'), ('간', 'Suffix'), ('전', 'Noun'), ('세계',
```

- 명사 추출 : okt.nouns( )

```
print(okt.nouns('지난 몇 달간 전 세계 모든 사람은 코로나19로 인해 ₩
전례 없는 고통을 겪으며 다양한 방식으로 심각하게 피해를 겪었습니다.'))
```

```
    ['지난', '몇', '달', '전', '세계', '모든', '사람', '코로나', '로', '전례', '고통', '방식', 'I
```

(3) Kkma 형태소 분석기

- 형태소(Morpheme)

```
from konlpy.tag import Kkma
```

- 토큰화 : kkma.morphs( )

```
kkma = Kkma()

print(kkma.morphs('지난 몇 달간 전 세계 모든 사람은 코로나19로 인해 ₩
전례 없는 고통을 겪으며 다양한 방식으로 심각하게 피해를 겪었습니다.'))
```

```
    ['지나', 'ㄴ', '몇', '달', '간', '전', '세계', '모든', '사람', '은', '코로나', '19', '로', '
```

- 품사 태깅 : kkma.pos( )

```
print(kkma.pos('지난 몇 달간 전 세계 모든 사람은 코로나19로 인해 ₩
전례 없는 고통을 겪으며 다양한 방식으로 심각하게 피해를 겪었습니다.'))
```

```
    [('지나', 'VV'), ('ㄴ', 'ETD'), ('몇', 'MDT'), ('달', 'NNG'), ('간', 'NNB'), ('전', 'NNG'), (
```

- 명사 추출 : kkma.nouns( )

```
print(kkma.nouns('지난 몇 달간 전 세계 모든 사람은 코로나19로 인해 ₩
전례 없는 고통을 겪으며 다양한 방식으로 심각하게 피해를 겪었습니다.'))
```

```
['달', '달간', '간', '전', '세계', '사람', '코로나', '코로나19', '19', '전례', '고통', '다양'
```

II. Encoding

1) Encoding with Keras

(1) Import Package

```
from keras.preprocessing.text import Tokenizer
from keras.utils import to_categorical
```

(2) 실습 문장

```
sentence = '가지마라 가지마라 그녀는 위험해 매력이 너무 넘치는 Girl ₩
하지마라 하지마라 사랑은 위험해 ₩
내가 내가 내가 먼저 네게 네게 네게 빠져 빠져 빠져 버려 baby'
```

2) 정수 인코딩(Integer Encoding)

(1) Tokenizer.fit_on_texts( )

- Tokenization & Integer Indexing

```
from keras.preprocessing.text import Tokenizer

tknz = Tokenizer()
tknz.fit_on_texts([sentence])
```

```
print(tknz.word_index)
```

```
{'내가': 1, '네게': 2, '빠져': 3, '가지마라': 4, '위험해': 5, '하지마라': 6, '그녀는': 7, '매
```

(2) Tokenizer.texts_to_sequences( )

- Integer Encoding

```
LBE = tknz.texts_to_sequences([sentence])
```

```
print(LBE)
```

```
[[4, 4, 7, 5, 8, 9, 10, 11, 6, 6, 12, 5, 1, 1, 1, 13, 2, 2, 2, 3, 3, 3, 14, 15]]
```

3) 원-핫 인코딩(One-Hot Encoding)

(1) to_categorical( )

- One-Hot Encoding

```
from keras.utils import to_categorical

OHE = to_categorical(LBE)
```

```
print(OHE)
```

```
[[[0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
  [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
  [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]]
```

#

#

#

The End

#

#

#

✓ 0초 오후 1:48에 완료됨 ● ×