Text_cleaning (1)

August 19, 2022

0.1 Text Cleanup

[1]:
```
!python -m spacy download en_core_web_sm
```

```
  File "/usr/local/lib/python3.7/site.py", line 177
    file=sys.stderr)
        ^
SyntaxError: invalid syntax
```

[2]:
```
text = 'I am learning Natural Language Processing. I am fond of it. I like to␣
↪make my career in the same field.'
text
```

[2]:
```
'I am learning Natural Language Processing. I am fond of it. I like to make my
career in the same field.'
```

[3]:
```
## Importing the nltk library

import nltk
```

0.1.1 Sentence Segmentation

[4]:
```
nltk.tokenize.sent_tokenize(text)
```

[4]:
```
['I am learning Natural Language Processing.',
 'I am fond of it.',
 'I like to make my career in the same field.']
```

[5]:
```
nltk.tokenize.sent_tokenize('My email address is waj@simplilearn.com. I live in␣
↪Hyderabad')
```

[5]:
```
['My email address is waj@simplilearn.com.', 'I live in Hyderabad']
```

0.1.2 Word Tokenization

[6]:
```
for sentences in nltk.tokenize.sent_tokenize(text):
    print(sentences,' : ',nltk.tokenize.word_tokenize(sentences))
```

```
I am learning Natural Language Processing.  :  ['I', 'am', 'learning',
'Natural', 'Language', 'Processing', '.']
I am fond of it.  :  ['I', 'am', 'fond', 'of', 'it', '.']
I like to make my career in the same field.  :  ['I', 'like', 'to', 'make',
'my', 'career', 'in', 'the', 'same', 'field', '.']
```

[7]:
```
## Limitations with nltk word tokenizer
nltk.tokenize.word_tokenize('''Mr. Michael O`Neil works at Microsoft, located␣
 ↪at 45 Avenue, United States of America''')
```

[7]:
```
['Mr.',
 'Michael',
 'O',
 '`',
 'Neil',
 'works',
 'at',
 'Microsoft',
 ',',
 'located',
 'at',
 '45',
 'Avenue',
 ',',
 'United',
 'States',
 'of',
 'America']
```

[8]:
```
### type of tokenizer

# nltk.tokenize.TweetTokenizer
```

[9]:
```
text = 'I am working for the England cricket board as analytics engineer'

# Some words which are frequently used words, most common words such as I, am,␣
 ↪for, the, as
#Stopwords - words which do not add any meaning to the sentence

#Steps to remove stopwords, lowercase the text, word tokenize, filter it from␣
 ↪the stopword dictionary
```

0.1.3 Stopword Removal

[10]:
```
#Create your stopword dictionary
my_stopwords = ['i', 'am', 'for', 'the', 'as']
```

[11]:
```
# Lowercase, tokenize and filter from the above created dictionary
clean_text = [word for word in nltk.tokenize.word_tokenize(text.lower()) if␣
 ↪word not in my_stopwords]
clean_text
```

[11]:
```
['working', 'england', 'cricket', 'board', 'analytics', 'engineer']
```

[12]:
```
text = 'He is an experienced Natural Language Processing Engineer at Microsoft.␣
 ↪'

clean_text = [word for word in nltk.tokenize.word_tokenize(text.lower()) if␣
 ↪word not in my_stopwords]
clean_text
```

[12]:
```
['he',
 'is',
 'an',
 'experienced',
 'natural',
 'language',
 'processing',
 'engineer',
 'at',
 'microsoft',
 '.']
```

The stopwords are not exhaustive. Hence either we can keep updating our stopword list every time we encounter a stop word or use a previously available stop word list within nltk

[13]:
```
# Using pre-available list within nltk
from nltk.corpus import stopwords

mystopwords = stopwords.words('english')
print(mystopwords)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're",
"you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',
'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's",
'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what',
'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is',
'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
```

```
'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above',
'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under',
'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why',
'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some',
'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn',
"couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn',
"hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't",
'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
"shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]
```

[14]:
```
# Filtering using the nltk's stopword list

text = 'He is a knowledgeable Natural Language Processing Engineer.'

clean_text = [word for word in nltk.tokenize.word_tokenize(text.lower()) if␣
 ↪word not in mystopwords]
clean_text
```

[14]:
```
['knowledgeable', 'natural', 'language', 'processing', 'engineer', '.']
```

[15]:
```
text = 'He is having experience of 4+ years in the field of NLP. I am working␣
 ↪with him as his colleague since 2 years.'

clean_text = [word for word in nltk.tokenize.word_tokenize(text.lower()) if␣
 ↪word not in mystopwords]
clean_text
```

[15]:
```
['experience',
 '4+',
 'years',
 'field',
 'nlp',
 '.',
 'working',
 'colleague',
 'since',
 '2',
 'years',
 '.']
```

The punctuations and the digits are not removed by stopword removal. How to remove them?

0.1.4 Digits and Punctuation Removal

[16]:
```
from string import punctuation
```

[17]:
```
punctuation
```

[17]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

[18]:
```
'3'.isdigit()
```

[18]:
```
True
```

[19]:
```
text = 'He is having experience of 4+ years in the field of NLP. I am working␣
 ↪with him as his colleague since 2 years.'

clean_text = [word for word in nltk.tokenize.word_tokenize(text.lower()) if␣
 ↪word not in mystopwords]
# Removing punctuation from the stopword removed text
clean_text = [word for word in clean_text if word not in punctuation]
# Removing digits
clean_text = [word for word in clean_text if not word.isdigit()]
clean_text
```

[19]:
```
['experience',
 '4+',
 'years',
 'field',
 'nlp',
 'working',
 'colleague',
 'since',
 'years']
```

0.1.5 Stemming and Lemmatization

[20]:
```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
```

[21]:
```
stemmer.stem('cars')
```

[21]:
```
'car'
```

[22]:
```
stemmer.stem('policies')
```

[22]:
```
'polici'
```

[23]:
```
stemmer.stem('revolution')
```

[23]:
```
'revolut'
```

[24]:
```
stemmer.stem('better')
```

[24]:
```
'better'
```

[25]:
```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
```

[26]:
```
lemmatizer.lemmatize('better',pos='a')
```

[26]:
```
'good'
```

[27]:
```
lemmatizer.lemmatize('better')
```

[27]:
```
'better'
```

[28]:
```
#Implementing Stemming

text = 'He is having experience of 4+ years in the field of NLP. I am working␣
 ↪with him as his colleague since 2 years.'

clean_text = [word for word in nltk.tokenize.word_tokenize(text.lower()) if␣
 ↪word not in mystopwords]
# Removing punctuation from the stopword removed text
clean_text = [word for word in clean_text if word not in punctuation]
# Removing digits
clean_text = [word for word in clean_text if not word.isdigit()]
# Stemming of the text
clean_text = [stemmer.stem(word) for word in clean_text]
clean_text
```

[28]:
```
['experi', '4+', 'year', 'field', 'nlp', 'work', 'colleagu', 'sinc', 'year']
```

[29]:
```
#Implementing Lemmatization

text = 'He is having experience of 4+ years in the field of NLP. I am working␣
 ↪with him as his colleague since 2 years.'

clean_text = [word for word in nltk.tokenize.word_tokenize(text.lower()) if␣
 ↪word not in mystopwords]
# Removing punctuation from the stopword removed text
```

```
clean_text = [word for word in clean_text if word not in punctuation]
# Removing digits
clean_text = [word for word in clean_text if not word.isdigit()]
# Stemming of the text
clean_text = [lemmatizer.lemmatize(word,pos='v') for word in clean_text]
clean_text
```

[29]:
```
['experience',
 '4+',
 'years',
 'field',
 'nlp',
 'work',
 'colleague',
 'since',
 'years']
```

Part of Speech Tagging

[30]:
```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")

for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_, token.is_alpha,␣
 ↪token.is_stop, spacy.explain(token.pos_))
```

```
Apple Apple PROPN NNP True False proper noun
is be AUX VBZ True True auxiliary
looking look VERB VBG True False verb
at at ADP IN True True adposition
buying buy VERB VBG True False verb
U.K. U.K. PROPN NNP False False proper noun
startup startup NOUN NN True False noun
for for ADP IN True True adposition
$ $ SYM $ False False symbol
1 1 NUM CD False False numeral
billion billion NUM CD True False numeral
```

[31]:
```
[token.text for token in doc if token.is_stop==False]
```

[31]:
```
['Apple', 'looking', 'buying', 'U.K.', 'startup', '$', '1', 'billion']
```

Named Entity Recognition

[32]:
```
for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_, spacy.explain(ent.
 ↪label_))
```

```
Apple 0 5 ORG Companies, agencies, institutions, etc.
U.K. 27 31 GPE Countries, cities, states
$1 billion 44 54 MONEY Monetary values, including unit
```

[33]:
```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("An apple a day keeps a doctor away")

for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_, token.is_alpha,␣
 ↪token.is_stop, spacy.explain(token.pos_))
```

```
An an DET DT True True determiner
apple apple NOUN NN True False noun
a a DET DT True True determiner
day day NOUN NN True False noun
keeps keep VERB VBZ True False verb
a a DET DT True True determiner
doctor doctor NOUN NN True False noun
away away ADV RB True False adverb
```

[36]:
```
for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_, spacy.explain(ent.
 ↪label_))
```

[ ]:
```

```