Text_Preprocessing_1

June 19, 2022

1 Text Preprocessing - 1

Before building our nlp model, the textual data has to be cleaned up before converting them into vectors. According to how well the text is preprocessed, the models will exhibit its performance.
It involves several steps based on the data that is available to us:

1. Lower casing of the words
2. Removal of HTML tags
3. Removal of URLs
4. Removal of Punctuations
5. Chat Words Treatment
6. Emoji Treatment
7. Spelling Correction
8. Removal of Stopwords
9. Tokenization
10. Stemming and Lemmatization

The above steps completely depend on what kind of data available to us. Not all of the steps are compulsory. But removal of stopwords, tokenization, stemming or lemmatization is necessary for text preprocessing step.

To cover how all of the above steps will be carried out, we will be using two datasets taken from kaggle and custom sentences.

2 Importing Necessary Libraries

[1]:
```
####################--------- Basic Libraries ---------####################
import numpy as np
import pandas as pd

####################--------- NLP Libraries ---------####################
# Regular Expressions
import re

# For string manipulation
import string,time

# For emoji treatment
```

```
import emoji                                # pip install emoji

# For spelling correction
from textblob import TextBlob               # pip install -U textblob

# Nltk library
import nltk                                 # For installation -> https://www.nltk.org/
↪install.html

# list of stopwords
from nltk.corpus import stopwords

# For tokenization
from nltk.tokenize import word_tokenize, sent_tokenize

# For Stemming
from nltk.stem.porter import PorterStemmer

# For Lemmatization
from nltk.stem import WordNetLemmatizer

# Spacy library
import spacy                                # !pip install spacy && python -m spacy␣
↪download en
```

3 Loading the Dataset

Original source: - Disaster Tweets Dataset - https://www.kaggle.com/competitions/nlp-getting-started/data - IMDB Movie Reviews - https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

[2]:
```
disaster_tweet_df = pd.read_csv('./disaster_tweets.csv')
movies_review_df = pd.read_csv('./IMDB-Dataset.csv')
```

[3]:
```
disaster_tweet_df.sample(5)
```

[3]:

	id	keyword	location	text	target
2095	3011	death	New York	Xbox 360 Pro Console - *Red Ring of Death* - F…	0
4452	6335	hostages	THANJAVUR	2 hostages in Libya remain unharmed: Governmen…	1
5572	7952	rainstorm	North Vancouver, BC	Yay I can feel the wind gearing up for a rains…	1
5864	8377	ruin	NaN		
1670	2413	collide	Viejo		

```
5864           Don't let a few assholes ruin your night       0
1670      It's always super awkward when worlds collide       0
```

[4]:
```
movies_review_df.sample(5)
```

[4]:
```
                                                  review sentiment
25942  I found this movie on Netflix and had to add i…  positive
14700  Since the Little Mermaid was one of my favorit…  negative
5888   I'm a big fan of H. P. Lovecraft's books, and …  negative
47752  Someone, some day, should do a study of archit…  negative
29518  This movie was different in that it didn't sho…  negative
```

4 Lower casing of all the sentences

It is the first step carried out in text preprocessing as the text is required to be converted into the same case. But for some problems, this can also lead to loss of information e.g. in chats, upper case letters are used as emotions either to represent either they are angry or excited.

[5]:
```
disaster_tweet_df['text'] = disaster_tweet_df['text'].str.lower()
movies_review_df['review'] = movies_review_df['review'].str.lower()
```

[6]:
```
disaster_tweet_df['text']
```

[6]:
```
0       our deeds are the reason of this #earthquake m…
1                  forest fire near la ronge sask. canada
2       all residents asked to 'shelter in place' are …
3       13,000 people receive #wildfires evacuation or…
4       just got sent this photo from ruby #alaska as …
                              …
7608    two giant cranes holding a bridge collapse int…
7609    @aria_ahrary @thetawniest the out of control w…
7610    m1.94 [01:04 utc]?5km s of volcano hawaii. htt…
7611    police investigating after an e-bike collided …
7612      the latest: more homes razed by northern calif…
Name: text, Length: 7613, dtype: object
```

[7]:
```
movies_review_df['review']
```

[7]:
```
0          one of the other reviewers has mentioned that …
1          a wonderful little production. <br /><br />the…
2          i thought this was a wonderful way to spend ti…
3          basically there's a family where a little boy …
4          petter mattei's "love in the time of money" is…
                                 …
49995      i thought this movie did a down right good job…
49996      bad plot, bad dialogue, bad acting, idiotic di…
49997      i am a catholic taught in parochial elementary…
```

```
49998    i'm going to have to disagree with the previou…
49999    no one expects the star trek movies to be high…
Name: review, Length: 50000, dtype: object
```

5 Removal of HTML Tags

For nlp models, HTML tags are more of a junk, unnecessary data as it is meaningless. Therefore, it has to be removed.

[8]:
```
def remove_html_tags(text):
    pattern = re.compile('<.*?>')
    return pattern.sub(r'', text)
```

Before removal of HTML tags.

[9]:
```
movies_review_df['review'][1]
```

[9]:
```
'a wonderful little production. <br /><br />the filming technique is very
unassuming- very old-time-bbc fashion and gives a comforting, and sometimes
discomforting, sense of realism to the entire piece. <br /><br />the actors are
extremely well chosen- michael sheen not only "has got all the polari" but he
has all the voices down pat too! you can truly see the seamless editing guided
by the references to williams\' diary entries, not only is it well worth the
watching but it is a terrificly written and performed piece. a masterful
production about one of the great master\'s of comedy and his life. <br /><br
/>the realism really comes home with the little things: the fantasy of the guard
which, rather than use the traditional \'dream\' techniques remains solid then
disappears. it plays on our knowledge and our senses, particularly with the
scenes concerning orton and halliwell and the sets (particularly of their flat
with halliwell\'s murals decorating every surface) are terribly well done.'
```

[10]:
```
movies_review_df['review'] = movies_review_df['review'].apply(remove_html_tags)
```

After removal of HTML tags.

[11]:
```
movies_review_df['review'][1]
```

[11]:
```
'a wonderful little production. the filming technique is very unassuming- very
old-time-bbc fashion and gives a comforting, and sometimes discomforting, sense
of realism to the entire piece. the actors are extremely well chosen- michael
sheen not only "has got all the polari" but he has all the voices down pat too!
you can truly see the seamless editing guided by the references to williams\'
diary entries, not only is it well worth the watching but it is a terrificly
written and performed piece. a masterful production about one of the great
master\'s of comedy and his life. the realism really comes home with the little
things: the fantasy of the guard which, rather than use the traditional
\'dream\' techniques remains solid then disappears. it plays on our knowledge
and our senses, particularly with the scenes concerning orton and halliwell and
```

```
the sets (particularly of their flat with halliwell\'s murals decorating every
surface) are terribly well done.'
```

6 Removal of URLs

URLs have no semantic information in the text data. Therefore, it has to be removed.

Before removal of URLs.

[12]:
```
disaster_tweet_df[disaster_tweet_df['text'].str.contains('http')].head(5)
```

[12]:

	id	keyword	location	text	target
31	48	ablaze	Birmingham	@bbcmtd wholesale markets ablaze http://t.co/l…	1
32	49	ablaze	Est. September 2012 - Bristol	we always try to bring the heavy. #metal #rt h…	0
33	50	ablaze	AFRICA	#africanbaze: breaking news:nigeria flag set a…	1
35	53	ablaze	London, UK	on plus side look at the sky last night it was…	0
37	55	ablaze	World Wide!!	inec office in abia set ablaze - http://t.co/3…	1

[13]:
```
movies_review_df[movies_review_df['review'].str.contains('http')].head(5)
```

[13]:

	review	sentiment
907	following directly from where the story left o…	positive
1088	this quasi j-horror film followed a young woma…	negative
1972	the basic plot of 'marigold' boasts of a roman…	negative
2132	i, too, found "oppenheimer" to be a brilliant …	positive
3038	i really love this movie , i saw it for the fi…	positive

[14]:
```
def remove_url(text):
    pattern = re.compile(r'https?: ?//\S+|www\.\S+')
    return pattern.sub(r'', text)
```

[15]:
```
disaster_tweet_df['text'] = disaster_tweet_df['text'].apply(remove_url)
movies_review_df['review'] = movies_review_df['review'].apply(remove_url)
```

After removal of URLs.

[16]:
```
disaster_tweet_df[disaster_tweet_df['text'].str.contains('http')].head(5)
```

[16]:

	id	keyword	location	\
121	174	aftershock	Baker City Oregon	

```
                                                    text  target
121  aftershock: protect yourself and profit in the…       0
```

[17]:
```
movies_review_df[movies_review_df['review'].str.contains('http')].head(5)
```

[17]:
```
Empty DataFrame
Columns: [review, sentiment]
Index: []
```

Let's remove the text 'http' from disaster_tweet_df dataset.

[18]:
```
disaster_tweet_df['text'] = disaster_tweet_df['text'].str.replace('http', '')
```

[19]:
```
disaster_tweet_df[disaster_tweet_df['text'].str.contains('http')].head(5)
```

[19]:
```
Empty DataFrame
Columns: [id, keyword, location, text, target]
Index: []
```

7 Removal of Punctuation

Punctuation marks are not required as there is no effect of it on the model. It is just consuming more space. Therefore, it is important to remove the punctuation marks.

[20]:
```
string.punctuation
```

[20]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

[21]:
```
exclude = string.punctuation
```

First we will use basic function to remove the punctuation.

[22]:
```
def remove_punc(text):
    for char in exclude:
        text = text.replace(char,'')
    return text
```

[23]:
```
text = 'string. With. Punctuation?'
```

[24]:
```
start = time.time()
print(remove_punc(text))
time1 = time.time() - start
print(time1*50000)
```

```
string With Punctuation
8.344650268554688
```

We can see the code for removing the punctuation marks is very slow. Therefore, it is not recommended to use for large dataset. Fortunately, we have an alternative option for it.

[25]:
```
def remove_punc_2(text):
    return text.translate(str.maketrans('', '', exclude))
```

[26]:
```
start = time.time()
print(remove_punc_2(text))
time2 = time.time() - start
print(time2*50000)
```

```
string With Punctuation
9.560585021972656
```

Let's remove punctuation marks in our text datasets.

Before removal of punctuation marks.

[27]:
```
disaster_tweet_df.head()
```

[27]:
```
   id keyword location                                                text  \
0   1     NaN      NaN  our deeds are the reason of this #earthquake m…
1   4     NaN      NaN             forest fire near la ronge sask. canada
2   5     NaN      NaN  all residents asked to 'shelter in place' are …
3   6     NaN      NaN  13,000 people receive #wildfires evacuation or…
4   7     NaN      NaN  just got sent this photo from ruby #alaska as …

   target
0       1
1       1
2       1
3       1
4       1
```

[28]:
```
movies_review_df.head()
```

[28]:
```
                                              review sentiment
0  one of the other reviewers has mentioned that …  positive
1  a wonderful little production. the filming tec…  positive
2  i thought this was a wonderful way to spend ti…  positive
3  basically there's a family where a little boy …  negative
4  petter mattei's "love in the time of money" is…  positive
```

[29]:
```
disaster_tweet_df['text'] = disaster_tweet_df['text'].apply(remove_punc_2)
movies_review_df['review'] = movies_review_df['review'].apply(remove_punc_2)
```

After removal of punctuation marks.

[30]:
```
disaster_tweet_df.head()
```

[30]:
```
   id keyword location                                               text  \
0   1     NaN      NaN  our deeds are the reason of this earthquake ma…
1   4     NaN      NaN             forest fire near la ronge sask canada
```

```
2   5     NaN      NaN  all residents asked to shelter in place are be…
3   6     NaN      NaN  13000 people receive wildfires evacuation orde…
4   7     NaN      NaN  just got sent this photo from ruby alaska as s…
```

```
   target
0       1
1       1
2       1
3       1
4       1
```

[31]:
```
movies_review_df.head()
```

[31]:

	review	sentiment
0	one of the other reviewers has mentioned that …	positive
1	a wonderful little production the filming tech…	positive
2	i thought this was a wonderful way to spend ti…	positive
3	basically theres a family where a little boy j…	negative
4	petter matteis love in the time of money is a …	positive

8 Chat Word Treatment

Chat words cannot be understood by the machine. Therefore, we need to treat them into proper sentence for the machines to understand.

Source: https://github.com/rishabhverma17/sms_slang_translator/blob/master/slang.txt

[32]:
```
with open('slang.txt', 'r') as f:
    text = f.read()

list_of_slang = text.split("\n")

chat_words = {}
for sentence in list_of_slang:
    if sentence.find("=") != -1:
        key, value = sentence.split("=")[0], sentence.split("=")[1]
        chat_words[key] = value
```

[33]:
```
chat_words
```

[33]:
```
{'AFAIK': 'As Far As I Know',
 'AFK': 'Away From Keyboard',
 'ASAP': 'As Soon As Possible',
 'ATK': 'At The Keyboard',
 'ATM': 'At The Moment',
 'A3': 'Anytime, Anywhere, Anyplace',
 'BAK': 'Back At Keyboard',
 'BBL': 'Be Back Later',
```

```
'BBS': 'Be Back Soon',
'BFN': 'Bye For Now',
'B4N': 'Bye For Now',
'BRB': 'Be Right Back',
'BRT': 'Be Right There',
'BTW': 'By The Way',
'B4': 'Before',
'CU': 'See You',
'CUL8R': 'See You Later',
'CYA': 'See You',
'FAQ': 'Frequently Asked Questions',
'FC': 'Fingers Crossed',
'FWIW': "For What It's Worth",
'FYI': 'For Your Information',
'GAL': 'Get A Life',
'GG': 'Good Game',
'GN': 'Good Night',
'GMTA': 'Great Minds Think Alike',
'GR8': 'Great!',
'G9': 'Genius',
'IC': 'I See',
'ICQ': 'I Seek you (also a chat program)',
'ILU': 'ILU: I Love You',
'IMHO': 'In My Honest/Humble Opinion',
'IMO': 'In My Opinion',
'IOW': 'In Other Words',
'IRL': 'In Real Life',
'KISS': 'Keep It Simple, Stupid',
'LDR': 'Long Distance Relationship',
'LMAO': 'Laugh My A.. Off',
'LOL': 'Laughing Out Loud',
'LTNS': 'Long Time No See',
'L8R': 'Later',
'MTE': 'My Thoughts Exactly',
'M8': 'Mate',
'NRN': 'No Reply Necessary',
'OIC': 'Oh I See',
'PITA': 'Pain In The A..',
'PRT': 'Party',
'PRW': 'Parents Are Watching',
'ROFL': 'Rolling On The Floor Laughing',
'ROFLOL': 'Rolling On The Floor Laughing Out Loud',
'ROTFLMAO': 'Rolling On The Floor Laughing My A.. Off',
'SK8': 'Skate',
'STATS': 'Your sex and age',
'ASL': 'Age, Sex, Location',
'THX': 'Thank You',
```

```
'TTFN': 'Ta-Ta For Now!',
'TTYL': 'Talk To You Later',
'U': 'You',
'U2': 'You Too',
'U4E': 'Yours For Ever',
'WB': 'Welcome Back',
'WTF': 'What The F…',
'WTG': 'Way To Go!',
'WUF': 'Where Are You From?',
'W8': 'Wait…',
'7K': 'Sick:-D Laugher'}
```

[34]:
```
def chat_conversion(text):
    new_text = []
    for w in text.split():
        if w.upper() in chat_words:
            new_text.append(chat_words[w.upper()])
        else:
            new_text.append(w)
    return " ".join(new_text)
```

[35]:
```
chat_conversion('IMHO he is the best')
```

[35]:
```
'In My Honest/Humble Opinion he is the best'
```

[36]:
```
chat_conversion('FYI dhaka is the capital of bangladesh')
```

[36]:
```
'For Your Information dhaka is the capital of bangladesh'
```

9 Emojis Treatment

Emojis are also not understood by the machine. Therefore, it would be better to remove them or convert them into meaningful information.

9.1 Remove Emoji(s)

[37]:
```
def remove_emoji(text):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)
```

[38]:
```
remove_emoji("Loved the movie. It was   ")
```

[38]:
```
'Loved the movie. It was '
```

[39]:
```
remove_emoji("Lmao  ")
```

[39]:
```
'Lmao '
```

9.2 Converting Emojis into meaningful information

`.demojize` method from emoji library allows conversion of emoji into its description.

[40]:
```
print(emoji.demojize('Python is  '))
```

```
Python is :fire:
```

[41]:
```
print(emoji.demojize("Loved the movie. It was   "))
```

```
Loved the movie. It was :smiling_face_with_heart-eyes::smiling_face_with_heart-
eyes::smiling_face_with_heart-eyes:
```

10 Spelling Correction

This is to avoid complexity that will raise due to both words having different spellings. It can specially occur during tokenization.

[42]:
```
incorrect_text = 'ceertain conditionas duriing seveal ggenerations aree␣
↪moodified in the saame maner.'
```

[43]:
```
def spelling_corrector(text):
    textBlb = TextBlob(text)
    return textBlb.correct().string
```

[44]:
```
spelling_corrector(incorrect_text)
```

[44]:
```
'certain conditions during several generations are modified in the same manner.'
```

11 Removing Stopwords

These are the words that help in contributing sentence formation but do not contribute in the meaning of the sentence. Library `nltk` provides list of stopwords. But there are many words that are important that are included in stopwords which can completely change meaning of a sentence and can lead to loss of information that is important.

E.g. * Sentence 1 - The food is good. * Sentence 2 - The food is not good.

If we remove the stopwords from the above sentences then both of them will have same words and will be interpreted wrongly by the model. Therefore, it is better to make use of customized stopwords list.

[45]:
```
from nltk.corpus import stopwords
stopwords = stopwords.words('english')
```

[46]:
```
print(stopwords)
```

```
['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"]
```

Some of the words has been handpicked to be removed from stopwords list as it is very possible that they can change meaning of sentence(s).

[47]:
```
words_discard_from_stopwords = ['because', 'about', 'against', 'between',
                                'no', 'nor', 'not',  'before', 'after',
                                'do', 'during', 'above', 'below', 'over',␣
 ↪'under',
                                'further', 'once', 'how', 'all', 'any',
                                'don', "don't", 'should', '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",'mightn', "mightn't",␣
↪'mustn',
                                "mustn't", 'needn', "needn't", 'shouldn',␣
↪"shouldn't",
                                'wasn', "wasn't", 'weren', "weren't", 'won',␣
↪"won't",
                                'wouldn', "wouldn't"]
```

[48]:
```
for word in words_discard_from_stopwords:
    stopwords.remove(word)
```

[49]:
```
print(stopwords)
```

```
['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',
'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'as',
'until', 'while', 'of', 'at', 'by', 'for', 'with', 'into', 'through', 'to',
'from', 'up', 'down', 'in', 'out', 'on', 'off', 'again', 'then', 'here',
'there', 'when', 'where', 'why', 'both', 'each', 'few', 'more', 'most', 'other',
'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't',
'can', 'will', 'just', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y',
'ma', 'shan', "shan't"]
```

Now we are ready to remove stopwords from our datasets.

[50]:
```
disaster_tweet_df.head(5)
```

[50]:

	id	keyword	location	text	target
0	1	NaN	NaN	our deeds are the reason of this earthquake ma…	1
1	4	NaN	NaN	forest fire near la ronge sask canada	1
2	5	NaN	NaN	all residents asked to shelter in place are be…	1
3	6	NaN	NaN	13000 people receive wildfires evacuation orde…	1
4	7	NaN	NaN	just got sent this photo from ruby alaska as s…	1

[51]:
```
movies_review_df.head(3)
```

[51]:

	review	sentiment
0	one of the other reviewers has mentioned that …	positive
1	a wonderful little production the filming tech…	positive
2	i thought this was a wonderful way to spend ti…	positive

[52]:
```
def remove_stopwords(text):
    new_text = []

    for word in text.split():
```

```
        if word in stopwords:
            new_text.append('')
        else:
            new_text.append(word)
    x = new_text[:]
    new_text.clear()
    return " ".join(x)
```

[53]:
```
disaster_tweet_df['text'] = disaster_tweet_df['text'].apply(remove_stopwords)
movies_review_df['review'] = movies_review_df['review'].apply(remove_stopwords)
```

After removal of stopwords

[54]:
```
disaster_tweet_df.head(5)
```

[54]:

	id	keyword	location	text	target
0	1	NaN	NaN	deeds reason earthquake may allah forgive…	1
1	4	NaN	NaN	forest fire near la ronge sask canada	1
2	5	NaN	NaN	all residents asked shelter place notified…	1
3	6	NaN	NaN	13000 people receive wildfires evacuation orde…	1
4	7	NaN	NaN	got sent photo ruby alaska smoke wildfire…	1

[55]:
```
movies_review_df.head(3)
```

[55]:

	review	sentiment
0	one reviewers mentioned after watching 1…	positive
1	wonderful little production filming techniqu…	positive
2	thought wonderful way spend time hot s…	positive

12 Tokenization

It is the process of breaking your text document into smaller parts which is known as tokens. Two types of tokenization: - Sentence - Word

In this notebook, we will be using `nltk` and `spacy` library. These libraries will provide us built-in methods for tokenizing paragraphs into sentences and sentences into words.

Using NLTK methods

[56]:
```
paragraph = """Lorem Ipsum is simply dummy text of the printing and typesetting␣
 ↪industry?
Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
```

```
when an unknown printer took a galley of type and scrambled it to make a type␣
 ↪specimen book."""
```

[57]:
```
sent_tokenize(paragraph)
```

[57]:
```
['Lorem Ipsum is simply dummy text of the printing and typesetting industry?',
 "Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
\nwhen an unknown printer took a galley of type and scrambled it to make a type
specimen book."]
```

[58]:
```
sent_1 = 'I am going to Toronto, Canada!'
word_tokenize(sent_1)
```

[58]:
```
['I', 'am', 'going', 'to', 'Toronto', ',', 'Canada', '!']
```

Using Spacy methods

[59]:
```
nlp = spacy.load("en_core_web_sm")

tokenizer_spacy = lambda text : [token for token in nlp(text)]
```

[60]:
```
tokenizer_spacy(sent_1)
```

[60]:
```
[I, am, going, to, Toronto, ,, Canada, !]
```

Let's try both of the methods on the disaster tweets dataset.

[61]:
```
disaster_tweet_df['text'].apply(word_tokenize)
```

[61]:
```
0       [deeds, reason, earthquake, may, allah, forgiv…
1           [forest, fire, near, la, ronge, sask, canada]
2       [all, residents, asked, shelter, place, notifi…
3       [13000, people, receive, wildfires, evacuation…
4       [got, sent, photo, ruby, alaska, smoke, wildfi…
                              …
7608    [two, giant, cranes, holding, bridge, collapse…
7609    [ariaahrary, thetawniest, control, wild, fires…
7610                [m194, 0104, utc5km, volcano, hawaii]
7611    [police, investigating, after, ebike, collided…
7612    [latest, homes, razed, northern, california, w…
Name: text, Length: 7613, dtype: object
```

[62]:
```
disaster_tweet_df['text'].apply(tokenizer_spacy)
```

[62]:
```
0       [ , deeds,   , reason,   , earthquake, may, al…
1           [forest, fire, near, la, ronge, sask, canada]
2       [all, residents, asked,  , shelter,  , place, …
3       [13000, people, receive, wildfires, evacuation…
4       [ , got, sent,  , photo,  , ruby, alaska,  , s…
```

```
                                   …
7608    [two, giant, cranes, holding,  , bridge, colla…
7609    [ariaahrary, thetawniest,    , control, wild, …
7610          [m194, 0104, utc5, km,   , volcano, hawaii]
7611    [police, investigating, after,  , ebike, colli…
7612    [ , latest,  , homes, razed,  , northern, cali…
Name: text, Length: 7613, dtype: object
```

Thought `spacy` method is performing better `nltk` method, it is much slower. Therefore, if speed is priority, `nltk` library has to be preferred. So we will be tokenizing using `nltk` method for both the datasets.

[63]:
```
disaster_tweet_df['text'] = disaster_tweet_df['text'].apply(word_tokenize)
movies_review_df['review'] = movies_review_df['review'].apply(word_tokenize)
```

[64]:
```
disaster_tweet_df.head(5)
```

[64]:
```
   id keyword location                                               text  \
0   1     NaN      NaN  [deeds, reason, earthquake, may, allah, forgiv…
1   4     NaN      NaN      [forest, fire, near, la, ronge, sask, canada]
2   5     NaN      NaN  [all, residents, asked, shelter, place, notifi…
3   6     NaN      NaN  [13000, people, receive, wildfires, evacuation…
4   7     NaN      NaN  [got, sent, photo, ruby, alaska, smoke, wildfi…

   target
0       1
1       1
2       1
3       1
4       1
```

[65]:
```
movies_review_df.head(5)
```

[65]:
```
                                              review sentiment
0  [one, reviewers, mentioned, after, watching, 1…  positive
1  [wonderful, little, production, filming, techn…  positive
2  [thought, wonderful, way, spend, time, hot, su…  positive
3  [basically, theres, family, little, boy, jake,…  negative
4  [petter, matteis, love, time, money, visually,…  positive
```

[66]:
```
disaster_tweet_df['text'][0]
```

[66]:
```
['deeds', 'reason', 'earthquake', 'may', 'allah', 'forgive', 'us', 'all']
```

[67]:
```
movies_review_df['review'][0]
```

[67]:
```
['one',
 'reviewers',
```

```
'mentioned',
'after',
'watching',
'1',
'oz',
'episode',
'youll',
'hooked',
'right',
'exactly',
'happened',
'methe',
'first',
'thing',
'struck',
'about',
'oz',
'brutality',
'unflinching',
'scenes',
'violence',
'set',
'right',
'word',
'go',
'trust',
'not',
'show',
'faint',
'hearted',
'timid',
'show',
'pulls',
'no',
'punches',
'regards',
'drugs',
'sex',
'violence',
'hardcore',
'classic',
'use',
'wordit',
'called',
'oz',
'nickname',
'given',
```

```
'oswald',
'maximum',
'security',
'state',
'penitentary',
'focuses',
'mainly',
'emerald',
'city',
'experimental',
'section',
'prison',
'all',
'cells',
'glass',
'fronts',
'face',
'inwards',
'privacy',
'not',
'high',
'agenda',
'em',
'city',
'home',
'manyaryans',
'muslims',
'gangstas',
'latinos',
'christians',
'italians',
'irish',
'moreso',
'scuffles',
'death',
'stares',
'dodgy',
'dealings',
'shady',
'agreements',
'never',
'far',
'awayi',
'would',
'say',
'main',
'appeal',
```

```
'show',
'due',
'fact',
'goes',
'shows',
'wouldnt',
'dare',
'forget',
'pretty',
'pictures',
'painted',
'mainstream',
'audiences',
'forget',
'charm',
'forget',
'romanceoz',
'doesnt',
'mess',
'around',
'first',
'episode',
'ever',
'saw',
'struck',
'nasty',
'surreal',
'couldnt',
'say',
'ready',
'watched',
'developed',
'taste',
'oz',
'got',
'accustomed',
'high',
'levels',
'graphic',
'violence',
'not',
'violence',
'injustice',
'crooked',
'guards',
'wholl',
'sold',
```

```
'nickel',
'inmates',
'wholl',
'kill',
'order',
'get',
'away',
'well',
'mannered',
'middle',
'class',
'inmates',
'turned',
'prison',
'bitches',
'due',
'lack',
'street',
'skills',
'prison',
'experience',
'watching',
'oz',
'may',
'become',
'comfortable',
'uncomfortable',
'viewingthats',
'get',
'touch',
'darker',
'side']
```

The sentences in both the datasets has been successfully tokenized. We can also tokenize using Tensorflow library - https://www.tensorflow.org/text/guide/tokenizers

13 Stemming and Lemmatization

Stemming is a way to bring back a word in its original form. e.g. walking -> walk, walks -> walk. It is mainly used in information retrieval systems.

There are two ways to do stemming: - Porter Stemming - Snow ball Stemming

[68]:
```
ps = PorterStemmer()
```

[69]:
```
def stem_words(tokenized_text):
    return [ps.stem(word) for word in tokenized_text]
```

```
[70]: disaster_tweet_df['text'][0]
```

```
[70]: ['deeds', 'reason', 'earthquake', 'may', 'allah', 'forgive', 'us', 'all']
```

```
[71]: stem_words(disaster_tweet_df['text'][0])
```

```
[71]: ['deed', 'reason', 'earthquak', 'may', 'allah', 'forgiv', 'us', 'all']
```

```
[72]: stem_words(movies_review_df['review'][0])
```

```
[72]: ['one',
       'review',
       'mention',
       'after',
       'watch',
       '1',
       'oz',
       'episod',
       'youll',
       'hook',
       'right',
       'exactli',
       'happen',
       'meth',
       'first',
       'thing',
       'struck',
       'about',
       'oz',
       'brutal',
       'unflinch',
       'scene',
       'violenc',
       'set',
       'right',
       'word',
       'go',
       'trust',
       'not',
       'show',
       'faint',
       'heart',
       'timid',
       'show',
       'pull',
       'no',
       'punch',
```

```
'regard',
'drug',
'sex',
'violenc',
'hardcor',
'classic',
'use',
'wordit',
'call',
'oz',
'nicknam',
'given',
'oswald',
'maximum',
'secur',
'state',
'penitentari',
'focus',
'mainli',
'emerald',
'citi',
'experiment',
'section',
'prison',
'all',
'cell',
'glass',
'front',
'face',
'inward',
'privaci',
'not',
'high',
'agenda',
'em',
'citi',
'home',
'manyaryan',
'muslim',
'gangsta',
'latino',
'christian',
'italian',
'irish',
'moreso',
'scuffl',
'death',
```

```
'stare',
'dodgi',
'deal',
'shadi',
'agreement',
'never',
'far',
'awayi',
'would',
'say',
'main',
'appeal',
'show',
'due',
'fact',
'goe',
'show',
'wouldnt',
'dare',
'forget',
'pretti',
'pictur',
'paint',
'mainstream',
'audienc',
'forget',
'charm',
'forget',
'romanceoz',
'doesnt',
'mess',
'around',
'first',
'episod',
'ever',
'saw',
'struck',
'nasti',
'surreal',
'couldnt',
'say',
'readi',
'watch',
'develop',
'tast',
'oz',
'got',
```

```
'accustom',
'high',
'level',
'graphic',
'violenc',
'not',
'violenc',
'injustic',
'crook',
'guard',
'wholl',
'sold',
'nickel',
'inmat',
'wholl',
'kill',
'order',
'get',
'away',
'well',
'manner',
'middl',
'class',
'inmat',
'turn',
'prison',
'bitch',
'due',
'lack',
'street',
'skill',
'prison',
'experi',
'watch',
'oz',
'may',
'becom',
'comfort',
'uncomfort',
'viewingthat',
'get',
'touch',
'darker',
'side']
```

Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language.

Eventhough stemming is correctly taking place, the words doesn't make much sense. Therefore, we will proceed with lemmatization.

Lemmatization reduces the inflected words properly ensuring that the root word belongs to the language. In lemmatization, root word is called Lemma. A lemma is the canonical form, dictionary form, or citation form of a set of words.

[73]:
```
wordnet_lemmatizer = WordNetLemmatizer()
```

[74]:
```
def lemmatize_words(tokenized_text):
    return [wordnet_lemmatizer.lemmatize(word)for word in tokenized_text]
```

[75]:
```
lemmatize_words(disaster_tweet_df['text'][0])
```

[75]:
```
['deed', 'reason', 'earthquake', 'may', 'allah', 'forgive', 'u', 'all']
```

We can see that just for one sentence, it took a lot of time for lemmatizing, as behind the scenes, there are lot of comparisons carried out for giving correct word output.

If speed is priority, then stemming is the choice as it will perform a lot faster than lemmatization. Therefore, we will be applying stemming to both our datasets.

[76]:
```
disaster_tweet_df['text'] = disaster_tweet_df['text'].apply(stem_words)
movies_review_df['review'] = movies_review_df['review'].apply(stem_words)
```

[77]:
```
disaster_tweet_df.head()
```

[77]:

	id	keyword	location	text	target
0	1	NaN	NaN	[deed, reason, earthquak, may, allah, forgiv, …	1
1	4	NaN	NaN	[forest, fire, near, la, rong, sask, canada]	1
2	5	NaN	NaN	[all, resid, ask, shelter, place, notifi, offi…	1
3	6	NaN	NaN	[13000, peopl, receiv, wildfir, evacu, order, …	1
4	7	NaN	NaN	[got, sent, photo, rubi, alaska, smoke, wildfi…	1

[78]:
```
movies_review_df.head()
```

[78]:

	review	sentiment
0	[one, review, mention, after, watch, 1, oz, ep…	positive
1	[wonder, littl, product, film, techniqu, unass…	positive
2	[thought, wonder, way, spend, time, hot, summe…	positive
3	[basic, there, famili, littl, boy, jake, think…	negative
4	[petter, mattei, love, time, money, visual, st…	positive

Now that our dataset is cleaned up, they are ready to be converted into vectors.