Text PreProcessing

Dataset:

https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

Current Directory

In [1]:
```
import os
import pandas as pd
os.getcwd()
```

Out[1]:
```
'/content'
```

Mounting Drive

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
ount("/content/drive", force_remount=True).
```

Google drive link for the data

In [3]:
```
data_path = "/content/drive/MyDrive/FSDS_Job_Gurantee/NLP/IMDB Dataset.csv"
```

Reading data

In [4]:
```
df = pd.read_csv(data_path)
```

Shape of the data

In [5]:
```
df.shape
```

Out[5]:
```
(50000, 2)
```

Top 1000 rows of the data

In [6]:
```
df = df.head(1000)
```

Shape of the data

In [7]:
```
df.shape
```

Out[7]:
```
(1000, 2)
```

Top 5 rows of the data

In [8]:
```
df.head()
```

Out[8]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	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

Text Cleaning

Lower Case

- Convert all the text to lowercase to ensure consistency. For example, "Hello" becomes "hello"

In [9]:
```
df['review'][3]
```

Out[9]:
```
"Basically there's a family where a little boy (Jake) thinks there's a zombie in his
closet & his parents are fighting all the time.<br /><br />This movie is slower than
a soap opera... and suddenly, Jake decides to become Rambo and kill the zombie.<br />
<br />OK, first of all when you're going to make a film you must Decide if its a thri
ller or a drama! As a drama the movie is watchable. Parents are divorcing & arguing l
ike in real life. And then we have Jake with his closet which totally ruins all the f
ilm! I expected to see a BOOGEYMAN similar movie, and instead i watched a drama with
some meaningless thriller spots.<br /><br />3 out of 10 just for the well playing par
ents & descent dialogs. As for the shots with Jake: just ignore them."
```

In [10]:
```
df['review'] = df['review'].str.lower()
```

In [11]:
```
df
```

Out[11]:

	review	sentiment
0	one of the other reviewers has mentioned that ...	positive
1	a wonderful little production. the...	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
...	...	...
995	nothing is sacred. just ask ernie fosselius. t...	positive
996	i hated it. i hate self-aware pretentious inan...	negative
997	i usually try to be professional and construct...	negative
998	if you like me is going to see this in a film ...	negative
999	this is like a zoology textbook, given that it...	negative

1000 rows × 2 columns

Remove html tags

- Remove any HTML tags from the text using regular expressions or an HTML parser. For example, "
  Hello

  " becomes "Hello"

In [12]:
```
import re
def remove_html_tags(text):
    pattern = re.compile('<.*?>')
    return pattern.sub(r'', text)
```

In [13]:
```
text = "<html><body><p> Movie 1</p><p> Actor - Aamir Khan</p><p> Click here to <a href
```

In [14]:
```
remove_html_tags(text)
```

Out[14]:
```
' Movie 1 Actor - Aamir Khan Click here to download'
```

In [15]:
```
df['review'] = df['review'].apply(remove_html_tags)
```

In [16]:
```
df['review'][5]
```

Out[16]:
```
'probably my all-time favorite movie, a story of selflessness, sacrifice and dedicati
on to a noble cause, but it\'s not preachy or boring. it just never gets old, despite
my having seen it some 15 or more times in the last 25 years. paul lukas\' performanc
e brings tears to my eyes, and bette davis, in one of her very few truly sympathetic
roles, is a delight. the kids are, as grandma says, more like "dressed-up midgets" th
an children, but that only makes them more fun to watch. and the mother\'s slow awake
ning to what\'s happening in the world and under her own roof is believable and start
ling. if i had a dozen thumbs, they\'d all be "up" for this movie.'
```

Remove url

- Remove any URLs from the text using regular expressions or string matching. For example, "Check out this link: https://example.com" becomes "Check out this link:"

In [17]:
```
def remove_url(text):
    pattern = re.compile(r'https?://\S+|www\.\S+')
    return pattern.sub(r'', text)
```

In [18]:
```
text1 = 'Check out my youtube https://www.youtube.com/dswithbappy'
text2 = 'Check out my linkedin https://www.linkedin.com/in/boktiarahmed73/'
text3 = 'Google search here www.google.com'
text4 = 'For data click https://www.kaggle.com/'
```

In [19]:
```
remove_url(text1)
```

Out[19]:
```
'Check out my youtube '
```

Punctuation handling

- Remove or handle punctuation marks in the text. This can involve removing them completely or replacing them with spaces. For example, "Hello!" becomes "Hello" or "Hello!" becomes "Hello "

In [20]:
```
import string,time
string.punctuation
```

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

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

Out[21]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

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

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

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

```
string With Punctuation
193.1905746459961
```

In [25]:
```
def remove_punc1(text):
    return text.translate(str.maketrans('', '', exclude))
```

In [26]:
```
start = time.time()
remove_punc1(text)
time2 = time.time() - start
print(time2*50000)
```

```
6.926059722900391
```

In [27]:
```
time1/time2
```

Out[27]:
```
27.89328743545611
```

In [28]:
```
df['review'][5]
```

Out[28]:
```
'probably my all-time favorite movie, a story of selflessness, sacrifice and dedicati
on to a noble cause, but it\'s not preachy or boring. it just never gets old, despite
my having seen it some 15 or more times in the last 25 years. paul lukas\' performanc
e brings tears to my eyes, and bette davis, in one of her very few truly sympathetic
roles, is a delight. the kids are, as grandma says, more like "dressed-up midgets" th
an children, but that only makes them more fun to watch. and the mother\'s slow awake
ning to what\'s happening in the world and under her own roof is believable and start
ling. if i had a dozen thumbs, they\'d all be "up" for this movie.'
```

In [29]:
```
remove_punc1(df['review'][5])
```

Out[29]:
```
'probably my alltime favorite movie a story of selflessness sacrifice and dedication
to a noble cause but its not preachy or boring it just never gets old despite my havi
ng seen it some 15 or more times in the last 25 years paul lukas performance brings t
ears to my eyes and bette davis in one of her very few truly sympathetic roles is a d
elight the kids are as grandma says more like dressedup midgets than children but tha
t only makes them more fun to watch and the mothers slow awakening to whats happening
in the world and under her own roof is believable and startling if i had a dozen thum
bs theyd all be up for this movie'
```

Chat conversion handling

- Deal with common chat-style conversions like replacing "u" with "you" or "lol" with "laughing out loud". This can be done using predefined mappings or regular expressions. For example, "OMG, u won!" becomes "Oh my God, you won!"

In [30]:
```
chat_words = {
    'AFAIK':'As Far As I Know',
    'AFK':'Away From Keyboard',
    'ASAP':'As Soon As Possible'
}
```

In [31]:
```
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)
```

In [32]:
```
chat_conversion('Do this work ASAP')
```

Out[32]:
```
'Do this work As Soon As Possible'
```

Incorrect text handling

- Correct common misspellings or abbreviations in the text. This can be done using predefined mappings or external libraries/tools. For example, "gr8" becomes "great" or "I luv it" becomes "I love it"

In [33]:
```
from textblob import TextBlob
```

In [34]:
```
incorrect_text = 'ceertain conditionas duriing seveal ggenerations aree moodified in t

textBlb = TextBlob(incorrect_text)

textBlb.correct().string
```

Out[34]:
```
'certain conditions during several generations are modified in the same manner.'
```

Stopwords

- Remove common words that do not carry significant meaning, such as articles, prepositions, and pronouns. This helps reduce noise in the text data. For example, "The cat is on the mat" becomes "cat mat"

In [35]:
```
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[35]:
```
True
```

In [36]:
```
stopwords.words('english')
```

Out[36]:
```
['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"]
```

In [37]:
```
def remove_stopwords(text):
    new_text = []

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

In [38]:
```
remove_stopwords('probably my all-time favorite movie, a story of selflessness, sacrif
```

Out[38]:
```
'probably  all-time favorite movie,  story  selflessness, sacrifice  dedication   nob
le cause,    preachy  boring.   never gets old, despite   seen   15   times'
```

In [39]:
```
df.head()
```

Out[39]:

	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

In [40]:
```
df['review'].apply(remove_stopwords)
```

Out[40]:
```
0      one    reviewers  mentioned   watching  1 oz e...
1       wonderful little production.  filming techniq...
2       thought    wonderful way  spend time    hot s...
3      basically there's  family   little boy (jake) ...
4      petter mattei's "love   time  money"   visuall...
                           ...
995    nothing  sacred.  ask ernie fosselius.  days, ...
996     hated it.  hate self-aware pretentious inanit...
997     usually try   professional  constructive   cr...
998      like   going  see    film history class  som...
999      like  zoology textbook, given   depiction  a...
Name: review, Length: 1000, dtype: object
```

Remove emoji

- Remove or handle emojis in the text. Emojis can be replaced with their corresponding textual representations or removed entirely. For example, "I'm feeling 😊" becomes "I'm feeling"

In [41]:
```
import re
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)
```

In [42]:
```
remove_emoji("Loved the movie. It was 😘😘")
```

Out[42]:
```
'Loved the movie. It was '
```

In [43]:
```
remove_emoji("Lmao 😂😂")
```

Out[43]:
```
'Lmao '
```

In [44]:
```
!pip install emoji
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: emoji in /usr/local/lib/python3.10/dist-packages (2.4.0)
```

In [45]:
```
import emoji
print(emoji.demojize('Python is 🔥'))
```

```
Python is :fire:
```

In [46]:
```
print(emoji.demojize('Loved the movie. It was 😘'))
```

```
Loved the movie. It was :face_blowing_a_kiss:
```

Tokenization

- Tokenization is the process of breaking down a text or document into individual units called tokens. These tokens can be words, sentences, or even smaller units like characters or subwords, depending on the requirements of the task at hand.
- Here are short definitions of tokenization along with examples:

    1. Word Tokenization: Splitting a text into individual words.
        - For example:
            - Input: "Hello, how are you?"
            - Output: ["Hello", ",", "how", "are", "you", "?"]
    2. Sentence Tokenization: Splitting a text into individual sentences.
        - For example:
            - Input: "I love pizza. Do you?"
            - Output: ["I love pizza.", "Do you?"]
    3. Character Tokenization: Breaking a text into individual characters.
        - For example:
            - Input: "Hello"
            - Output: ["H", "e", "l", "l", "o"]

4. Subword Tokenization: Splitting words into smaller meaningful units. This is useful for languages with complex word formations or for tasks like machine translation.
   - For example:
     - Input: "Unhappiness"
     - Output: ["Un", "happiness"]
1. Tokenization with Contractions: Handling contractions by splitting them into their constituent parts.
   - For example:
     - Input: "I'm happy"
     - Output: ["I", "'m", "happy"]
1. Tokenization with Emoticons or Symbols: Treating emoticons or symbols as separate tokens.
   - For example:
     - Input: "I'm feeling 😊"
     - Output: ["I'm", "feeling", "😊"]
1. Tokenization in NLP Libraries:
   - NLTK: Using the word_tokenize() or sent_tokenize() functions.
   - Spacy: Accessing tokens through the tokenization pipeline.
   - Transformers (Hugging Face): Utilizing tokenization methods specific to transformer models.

1. Using the split function

- Split the text into tokens using whitespace or specific delimiters. For example, "Hello, how are you?" can be tokenized into ["Hello,", "how", "are", "you?"]

In [47]:
```
# word tokenization
sent1 = 'I am going to delhi'
sent1.split()
```

Out[47]:
```
['I', 'am', 'going', 'to', 'delhi']
```

In [48]:
```
# sentence tokenization
sent2 = 'I am going to delhi. I will stay there for 3 days. Let\'s hope the trip to be
sent2.split('.')
```

Out[48]:
```
['I am going to delhi',
 ' I will stay there for 3 days',
 " Let's hope the trip to be great"]
```

In [49]:
```
# Problems with split function
sent3 = 'I am going to delhi!'
sent3.split()
```

Out[49]:
```
['I', 'am', 'going', 'to', 'delhi!']
```

In [50]:
```
sent4 = 'Where do think I should go? I have 3 day holiday'
sent4.split('.')
```

Out[50]:
```
['Where do think I should go? I have 3 day holiday']
```

2. Regular Expression

- Use regular expressions to define patterns for tokenization. For example, splitting text on whitespace or punctuation marks

In [51]:
```
import re
sent3 = 'I am going to delhi!'
tokens = re.findall("[\w']+", sent3)
tokens
```

Out[51]:
```
['I', 'am', 'going', 'to', 'delhi']
```

In [52]:
```
text = """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 specimer
sentences = re.compile('[.!?] ').split(text)
sentences
```

Out[52]:
```
['Lorem Ipsum is simply dummy text of the printing and typesetting industry',
 "\nLorem Ipsum has been the industry's standard dummy text ever since the 1500s, \nw
hen an unknown printer took a galley of type and scrambled it to make a type specimen
book."]
```

3. NLTK

- The Natural Language Toolkit (NLTK) is a popular Python library for natural language processing. It provides various tokenization methods, such as word_tokenize, sent_tokenize, and regexp_tokenize

In [53]:
```
from nltk.tokenize import word_tokenize,sent_tokenize
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[53]:
```
True
```

In [54]:
```
sent1 = 'I am going to visit delhi!'
word_tokenize(sent1)
```

Out[54]:
```
['I', 'am', 'going', 'to', 'visit', 'delhi', '!']
```

In [55]:
```
text = """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 specimer

sent_tokenize(text)
```

Out[55]:
```
['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, \nwhe
n an unknown printer took a galley of type and scrambled it to make a type specimen b
ook."]
```

In [56]:
```
sent5 = 'I have a Ph.D in A.I'
sent6 = "We're here to help! mail us at nks@gmail.com"
sent7 = 'A 5km ride cost $10.50'

word_tokenize(sent5)
```

Out[56]:
```
['I', 'have', 'a', 'Ph.D', 'in', 'A.I']
```

In [57]:
```
word_tokenize(sent6)
```

Out[57]:
```
['We',
 "'re",
 'here',
 'to',
 'help',
 '!',
 'mail',
 'us',
 'at',
 'nks',
 '@',
 'gmail.com']
```

In [58]:
```
word_tokenize(sent7)
```

Out[58]:
```
['A', '5km', 'ride', 'cost', '$', '10.50']
```

4. Spacy (good)

- Spacy is another powerful Python library for natural language processing. It provides tokenization capabilities along with other advanced features like named entity recognition and part-of-speech tagging

In [59]:
```
import spacy
nlp = spacy.load('en_core_web_sm')
```

In [60]:
```
doc1 = nlp(sent5)
doc2 = nlp(sent6)
doc3 = nlp(sent7)
doc4 = nlp(sent1)
```

In [61]:
```
doc4 = nlp(sent1)
doc4
```

Out[61]:
```
I am going to visit delhi!
```

In [62]:
```
for token in doc4:
    print(token)
```

```
I
am
going
to
visit
delhi
!
```

In [63]:
```
df.head()
```

Out[63]:

	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

Stemming

- Stemming is the process of reducing words to their base or root form by removing suffixes. For example, "running" and "runs" both stem to "run". Stemming helps in reducing the vocabulary size and grouping related words together
- Example:
  - Word: Running
    - Porter Stemmer Output: run
    - Snowball Stemmer Output: run
    - Lancaster Stemmer Output: run

In [64]:
```
from nltk.stem.porter import PorterStemmer
```

In [65]:
```
ps = PorterStemmer()
def stem_words(text):
    return " ".join([ps.stem(word) for word in text.split()])
```

In [66]:
```
sample = "walk walks walking walked"
stem_words(sample)
```

Out[66]:
```
'walk walk walk walk'
```

In [67]:
```
text = 'probably my alltime favorite movie a story of selflessness sacrifice and dedic
print(text)
```

```
probably my alltime favorite movie a story of selflessness sacrifice and dedication t
o a noble cause but its not preachy or boring it just never gets old despite my havin
g seen it some 15 or more times in the last 25 years paul lukas performance brings te
ars to my eyes and bette davis in one of her very few truly sympathetic roles is a de
light the kids are as grandma says more like dressedup midgets than children but that
only makes them more fun to watch and the mothers slow awakening to whats happening i
n the world and under her own roof is believable and startling if i had a dozen thumb
s theyd all be up for this movie
```

In [68]:
```
stem_words(text)
```

Out[68]:
```
'probabl my alltim favorit movi a stori of selfless sacrific and dedic to a nobl caus
but it not preachi or bore it just never get old despit my have seen it some 15 or mo
re time in the last 25 year paul luka perform bring tear to my eye and bett davi in o
ne of her veri few truli sympathet role is a delight the kid are as grandma say more
like dressedup midget than children but that onli make them more fun to watch and the
mother slow awaken to what happen in the world and under her own roof is believ and s
tartl if i had a dozen thumb theyd all be up for thi movi'
```

Lemmatization

- Lemmatization is the process of reducing words to their base form (lemma) based on their dictionary meaning. Unlike stemming, lemmatization considers the context and part of speech of the word. For example, the lemma of "running" is "run"
- Example:
  - Word: Running
    - Lemmatizer Output: run

In [69]:
```
import nltk
from nltk.stem import WordNetLemmatizer
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
wordnet_lemmatizer = WordNetLemmatizer()

sentence = "He was running and eating at same time. He has bad habit of swimming after
punctuations="?:!.,;"
sentence_words = nltk.word_tokenize(sentence)
for word in sentence_words:
    if word in punctuations:
        sentence_words.remove(word)

sentence_words
print("{0:20}{1:20}".format("Word","Lemma"))
for word in sentence_words:
    print ("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word,pos='v')))
```

```
Word                Lemma
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

He	He
was	be
running	run
and	and
eating	eat
at	at
same	same
time	time
He	He
has	have
bad	bad
habit	habit
of	of
swimming	swim
after	after
playing	play
long	long
hours	hours
in	in
the	the
Sun	Sun

The END